Numerical methods for cardiovascular problems: computational electrocardiology and fluid dynamics in moving domains

Ph. D. candidate: Lucia MIRABELLA
Matr. 710721

Supervisor: Prof. Alessandro VENEZIANI
Tutor: Prof. Alessandro VENEZIANI
Coordinator: Prof. Paolo BISCARI

Milano, 2010
To Tiziano

“No road is long with good company”

Turkish proverb
Images

4.1 Geometry reconstruction from medical images

4.1.1 Acquisition of medical images

4.1.2 Segmentation and geometry reconstruction

4.2 Registration algorithms

4.2.1 Non-rigid viscous fluid registration algorithm

4.2.2 Moving domain

4.3 Application to real cases

4.3.1 Moving aorta

4.3.2 Moving left ventricle

4.4 Validation of the image-based simulations

5 An insight on algorithms and implementation details

5.1 The LifeV Library

5.2 Monodomain preconditioner

5.3 Model Adaptivity

5.4 Fluid-Registration algorithm

5.5 Moving domain simulations

6 Conclusions

A List of parameters

A.1 Tissue models

A.2 Ionic models

A.2.1 Rogers-McCulloch model

A.2.2 Luo-Rudy model

Acknowledgements
Abstract

In this work we address the mathematical modeling and numerical simulation of cardiovascular phenomena, such as the propagation of the electrical signal in the heart tissue and the blood flow in large arteries. A strong motivation of this research activity comes from the interaction with the medical community that pinpoints the relevance and the complexity of medical issues occurring in the human circulatory system. The long term goal of this research is to set up a system of mathematical models, numerical techniques and software tools able to help the clinical practice by providing additional information so to support the medical decision process. Two important components to reach this goal are the integration between medical data and mathematical models and the promptness with whom patient-specific numerical results can be provided to medical doctors. This thesis should be regarded in this framework, focusing both on the development of accurate and efficient numerical methods for electrocardiology simulations, and on the integration of medical data on the motion of biological tissues within the numerical models and simulations.

We first introduce the main features of the cardiovascular system, describing with more details the anatomy, the physiology and some pathological behaviors of the heart. In particular we present the organization of the myocardial tissue and the microscopical and macroscopical mechanisms that make the electrical signal propagate, since a large part of the present work concerns the effective numerical simulation of this phenomenon. We then focus on the mathematical models that are commonly used to describe the electrical signal propagation in the tissue. More precisely, models for electrocardiology result from the coupling between a cell level model, which reproduces the sudden variation of the potential across a single cell membrane, and a tissue level model, which describes how the transmembrane potential propagates in the tissue. The transmembrane potential variation in time (called action potential) is due to chemical processes at microscopical level and can be modeled by exploiting equations of different complexity, according to the level of detail to be reached. In the present work we focus on the simple Rogers-McCulloch model and the more complex Luo-Rudy phase I model. One of the most accurate models for the action potential propagation in the tissue is the Bidomain model, which describes both the extracellular and the intracellular domain, and the Monodomain model, a simplified version of the Bidomain one. Due to its mathematical structure, the Bidomain model is expensive to be solved numerically while the Monodomain model is less expensive but do not reproduce accurately the action potential propagation in some cases. Therefore two novel numerical methods have been developed during this work and are reported in the thesis, after a general presentation on the state of the art of numerical simulations for electrocardiology. In
particular we first derive an ad-hoc preconditioner for the Bidomain, which is based on a proper reformulation of the Monodomain problem\footnote{This work has been developed in collaboration with M. Perego (among the others), during his ph. D. program at the Department of Mathematics of Politecnico di Milano, Italy.}. This preconditioner is proven to be optimal with respect to the mesh size parameter and 3D numerical tests show that it leads to important reductions (even about 50\%) of the CPU time required by the simulation, with respect to a standard strategy. The second technique we propose is a model adaptivity strategy, which consists in solving a hybrid model called Hybridomain, which corresponds to solve the Bidomain model in a partition of the computational domain and the Monodomain one in the remaining part. The choice of the partition is time dependent and is performed by means of a model error estimator which controls the difference between the solution of the Bidomain and the solution of Hybridomain model. The Bidomain is locally activated where the model error estimator is large and the Monodomain is solved elsewhere. 3D tests with LifeV library, both in healthy and unhealthy cases, show that this strategy, which can also be coupled to the use of the Monodomain preconditioner, allows to save computational time and to maintain a good accuracy of the results.

In the last part of the present work we focus on developing and testing a pipeline to include the actual motion of biological structures in the simulation. This technique has a general validity, and it has been successfully applied in this work to the simulation of blood fluid dynamics in moving vessels. In particular from a set of medical images of a specific region acquired at different time frames during the cardiac cycle, we extract the domain of interest and we track the motion of each point in time, using a proper registration algorithm. To this extent, a registration algorithm called non-rigid viscous-fluid registration is studied and implemented, obtaining promising preliminary results. The further step is the formulation of the problem equations in a moving domain framework. In the case of blood flow in arteries the natural choice is the Arbitrary Lagrangian Eulerian formulation (ALE). Finally the simulation of the interested phenomenon can be performed in a moving domain. This image-based motion strategy is applied to the modeling of blood flow in a patient-specific aortic arch geometry, comparing the results with a fixed-domain simulation. We also proposed a validation of the approach, through a comparison with the results of a more standard fluid-structure interaction algorithm. The technique is promising since the differences between the two computed solution fields are very small and the image-based motion numerical simulations require less computational effort than standard FSI strategies. The same pipeline can be applied to perform electrocardiology simulations in a moving heart, but the large displacements occurring in the heart make it necessary to devise ad hoc registration algorithms.

Finally, since a large part of the present work has been devoted to develop and test numerical approaches, the ideation and implementation of the algorithms constitutes an important component of this work. For this reason we report in a separate Chapter of the thesis the most interesting aspects of the implementation of each method previously introduced.
1 Introduction

Cardiovascular diseases represent the leading cause of mortality in Western countries and their relevance is increasing in developing countries as well [3]. This evidence motivates the interest of the scientific community in this topic and promotes the collaboration among medical doctors, applied mathematicians, physicists and engineers. The purposes of this partnership range from the design of more effective and accurate devices for clinical analysis, to the investigation of correlations between biological phenomena, to the prediction of a pathology development or the success of a surgery or therapy, useful to help clinicians in making their decisions.

Several research programs nowadays exploit this network of knowledge. In particular we mention the European projects Virtual Physiological Human Network of Excellence\(^1\) and Mathcard\(^2\) and the international project IUPS Physiome Project\(^3\) comprising, among the others, the Wellcome Trust Heart Physiome Project. Several works, in the context of patient-specific modeling and simulation of the cardiovascular system, have been published in the last 10 years.

The cardiovascular system is a complex structure, which consists of various organs interacting one with another in many different ways. A complete and accurate description of the entire system is at this time unfeasible for the complexity and heterogeneity of the phenomena involved. Nowadays, available models either focus with a great level of detail on a specific part of the system, or consider the interaction of different parts, each of them described in a simplified way. We mention in particular [7, 8, 53, 127, 151, 157], referring to them for further details on this topics.

In general, physical complexity is corresponded by complexity of the mathematical models and the algorithms required for the numerical simulations. Hence, ad hoc strategies need to be developed and tested to obtain the desired results with reasonable computational efforts. The relevance of these medical problems stimulates the scientific community to pursuing this goal. In particular, the possibility of supporting clinical practice with information obtained by numerical simulations demands for the improvement of mathematical models and the efficiency of numerical algorithms. Most importantly, the reliability of computational experiments needs to be carefully assessed by extensive validation against the medical evidence.

The present work can be regarded in this framework, since it concerns both the development of numerical methods to simulate efficiently the electrical activity of the heart

---

\(^1\)http://www.vph-noe.eu/
\(^2\)http://www.mathcard.eu/
\(^3\)http://www.physiome.org.nz/
and the application and validation of a method for the accurate simulation of dynamics (cardiac potential and blood flow) in moving domains. In particular, the latter is based on a strong integration of numerical methods and medical imaging of the region of interest and can actually reduce the computational efforts.

The thesis is organized as follows. In the present Chapter we report a macroscopic description of the cardiovascular system, focusing in particular on the heart. In Chapter 2 we present the mathematical models used to describe the electrical activity of the heart, including cell membrane models and tissue propagation models, like the Bidomain and the Monodomain models. We then report the Navier-Stokes equations which describe blood flow in arteries, both in fixed and in moving domains. We also introduce models for the heart and vessels tissue mechanics. In Chapter 3 we first summarize the state of the art on numerical approaches to simulate the propagation of the electrical stimulus in the heart. We then present two numerical methods that can be employed to reduce the computational time required by the electrocardiology simulations. In particular we propose a model-based preconditioner for the Bidomain model and a model adaptivity algorithm based on an a posteriori model error estimator, able to switch automatically between a simplified model (Monodomain) and an accurate one (Bidomain). We report numerical results that show the validity of the techniques. In Chapter 4 we describe a strategy to take into account the actual motion of the considered region of the cardiovascular system, when performing numerical simulations. The motion can be captured by medical imaging devices during the clinical practice. The approach is quite general and can be in principle applied both to electrocardiology simulations and to fluid dynamics simulations. In this context we study and implement a specific technique called non-rigid viscous fluid registration, to track in time the motion of a considered structure. We then apply this approach to blood fluid dynamics simulations in large vessels and we validate the strategy by comparing the results with those obtained with more standard fluid-structure interaction algorithms. Finally in Chapter 5 we address the details of the algorithms developed in the present work, reporting also some implementation details.

1.1 An overview on the cardiovascular system

In this Section we present the main anatomical features of the cardiovascular system. In Section 1.1.1 we describe its functioning at a macroscopical level. Since the main part of the present work has been devoted to the effective simulation of mathematical models for cardiac electrophysiology, in Section 1.1.2 we focus on the heart, describing its tissue at microscopic and mesoscopic levels. We introduce the mechanism responsible for the electrical activation of the tissue. We also mention the most common causes of failure of the cardiac electrical system. This introduction is based mainly on [11, 28, 30, 45, 53, 67, 98] where we refer the interested reader for more details.
1.1.1 Macroscopical description of the circulatory system

The cardiovascular system consists of heart, blood and blood vessels and its main role is to bring oxygen and nutrients to every living tissue in the body and to remove wastes disposed of by lungs and kidneys. The propulsion of blood is performed by the heart.

Heart as a pump

The heart is a hollow muscular organ, lying in a connective tissue space between the vertebral column and the sternum. It is completely enveloped in a membrane, the pericardium, that extends between the pleural cavities, the diaphragm, and the great vessels. The shape of the heart resembles a truncated cone. The upper part is called base of the heart and is anchored by the great vessels entering the heart at this location. The lowest part is called apex and is freely mobile in the pericardial sac. The heart is composed of two separate pumps, completely divided by the interventricular septum: a right heart that pumps blood through the lungs and a left heart that pumps blood through the peripheral organs. Each of these pumps is composed of an atrium and a ventricle. Each atrium is a weak primer pump for the ventricle, helping to move blood into it. The blood is then propelled by the ventricles to the vessels network. The contraction of the ventricular myocardium is called systole; the relaxation is called diastole.

The most part of the atria chamber wall presents a smooth surface while the rest of the atria and the ventricle chambers wall show muscular ridges (trabeculae carneae) protruding into the chambers. The four valves of the heart are anchored in dense fibrous connective tissue rings that lie nearly in a plane. Together with the connective tissue among them they form a unit, the so-called cardiac skeleton, to which the atria and ventricles are attached above and below (Figure 1.1). The cusps of the valves between atria and ventricles (atrioventricular valves) arise as double layers of endocardium from the cardiac skeleton. The free ends of the valve cusps are attached by tendinous threads (chordae tendineae) to the papillary muscles. These cone-shaped processes on the inner ventricular walls, together with the chordae tendineae, prevent the cusps from flapping back during ventricular contractions. A valve with three cusps (tricuspid valve) is situated between the right atrium and the right ventricle. A valve with two cusps (bicuspid valve or mitral valve) separates the left atrium and left ventricle. The semilunar valves are situated at the entrances to the pulmonary artery and the aorta. They prevent the blood from flowing back after a completed ventricular contraction. The pulmonary and aortic valves consist of three pockets of duplicated endocardium that project into the lumen with their inferior surfaces directed towards the heart. When the margins of the semilunar valves are tightly apposed, the corresponding exit valve is closed. As the pressure in the ventricles increases, the margins of the semilunar valves draw apart, and the exit valve opens.

During the first part of systole, the ventricular myocardium begins to contract. Because the atrioventricular valves are closed, and the semilunar valves not yet open, the intraventricular pressure rises rapidly with no change in volume. However, as soon as the pressure in the ventricles reaches the pressure in the aorta (about 120 mmHg) or the
1 Introduction

The pulmonary artery (about 20 mmHg), the semilunar valves open, and the ejection phase begins. During this phase the ventricle is maximally contracted, and a stroke volume of 70 ml of blood is ejected into the arteries at rest. The intraventricular pressure then again falls below arterial pressure, decelerates the flow and, when the flow reverses, the semilunar valves close again. Systole is followed by diastole, during which the myocardium relaxes and at first the atrioventricular valves remain closed and the volume inside the ventricles is unchanged (the so-called end-diastolic volume of about 70 ml). The pressure in the ventricles then falls below that in the atria, so that the atrioventricular valves open and blood flows from the atria into the ventricles. The driving force for this movement is the beginning atrial contraction, and the descent of the base of the heart, by which the base approaches the apex during the ejection phase, expanding the atria and thus sucking blood out of the veins. As the ventricular myocardium relaxes, the base again travels upward, and blood reaches the ventricles through the open atrioventricular valves (see Figure 1.2).

Blood in the vascular network

Blood is ejected from the heart in discrete pulses under relatively high pressure into the main arteries where it flows through a network of branching arteries of decreasing size to the arterioles and then the capillaries, where it delivers oxygen and nutrients to the tissues and removes carbon dioxide and catabolites. Blood is collected from the capillaries through merging venules and returns to the heart at low pressure through a network of veins.
1 Introduction

Figure 1.2: Pressure-Volume loop of the left ventricle (LV) during the cardiac cycle. Modified from http://ocw.tufts.edu/Content/50/lecturenotes/634463/634527

Functionally the circulatory system can be divided into two parts: a greater (systemic) and a lesser (pulmonary) circulation (Figure 1.3). The deoxygenated (venous) blood from the lower and upper body travels through the great venous trunks to the right atrium and then through the right ventricle and the pulmonary artery to the lungs (pulmonary circulation).
monary circulation). The blood is enriched with oxygen in the lungs and flows through the pulmonary veins back into the left atrium of the heart. From there it reaches the left ventricle, which pumps the blood through the aorta into the greater circulation (systemic circulation). The blood is distributed through the whole body in the larger and smaller arteries and eventually reaches the terminal vessels, the capillaries. After exchanging substances and gases in the tissues, the blood returns to the heart through the venous part of the systemic circulation.

Each blood circuit, systemic and pulmonary, is thus composed of three main compartments: arteries, capillaries and veins. The primary purpose of the arterial and venous vessels is to carry blood to and from the various tissues, while the microcirculatory compartment carries out the various exchange processes. The vascular networks present various branching or merging junctions, that generate vessels with widely different diameters and lengths. The arterial and venous systems are primarily bifurcating trees although there are numerous interconnections, anastomoses, at different places in the body. It is estimated that there are approximately twenty generations of bifurcations going from the heart to the most distant capillary beds in both the systemic and pulmonary circulation. The venous network is almost parallel to the arterial network, except in the skull. The major arteries and their corresponding veins often run together with nerves as a neurovascular bundle surrounded by dense connective tissue. Veins, unlike arteries, have valves that prevent backflow of blood away from the heart. The boundary between the large vessels and the microcirculation is characterized on the basis of the diameter of the vessels, the threshold being between 100 and 250 \( \mu \)m. The transition between arterial and venous system is blurred, since the structure and composition of the smallest arteries and arterioles and of the narrowest veins and venules are very similar. The diameter of capillaries, the smallest blood vessels, is about 5 \( \mu \)m, which is considerable smaller than the largest diameter of a red blood cell and the topology of capillary beds is very complex with many interconnections.

In this thesis we consider only examples of fluid-dynamics in medium and large arteries, like the aortic arch. In both cases blood can be assumed to be a homogeneous incompressible fluid.

The arterial wall has a complex structure, which depends also on the locations in the arterial tree. The walls of the large arteries have a circumferentially layered structure, whose most internal layer is called intima and is composed of endothelium, the middle one is called media and is composed itself of layers of smooth muscle cells interspersed with elastic lamellae, and the most external layer is called adventitia consisting mainly of loose connective tissue, with some smooth muscle cells. The mentioned layers are separated by elastic lamellae.

Arteries proximal to the heart, as the ones considered in this thesis, are known as elastic arteries. Their walls can be slightly thinner than more distal arteries with a smaller fraction of smooth muscle cells, making them more distensible. Due to this observation, the vessel wall can be modeled as an elastic medium, as described in Section 2.2.2.
1 Introduction

1.1.2 Heart anatomy and activity

Tissue description

The cardiac tissue can be distinguished in the atrial muscle, the ventricular muscle, the excitatory tissue and the conductive muscle fibers. The first two types of tissue, also called common myocardium, have contractile properties similar to the skeletal muscle but the contraction in the heart is much longer, while the specialized excitatory and conductive tissue show a week contraction. The main feature of the excitatory tissue is the automatic generation of rhythmical electrical discharge, while the main function of the conductive tissue is the fast propagation of the electrical signal to the common myocardium.

The heart wall consists of three layers of different thickness and structure: an inner serous coat (endocardium) the actual cardiac muscle (myocardium) and an outer serous coat (epicardium). Between the epicardium and the inner side of the pericardium lies a thin serous cavity, containing a fluid which allows the frictionless movement of the heart in the pericardial sac. The myocardium consists of striated cardiac muscle and is about 0.7 cm thick in the right ventricle, while the atrial wall is thinner. The wall of the left ventricle, because of its higher pressure and the consequently increased load, averages about 1.4 cm in thickness.

At the microscopic level, cardiac muscle is a composite tissue. It consists of various cell types, mainly myocytes and fibroblasts, supported by an extra-cellular matrix (ECM) and permeated by fluids.

Cardiac myocytes are the major constituent of heart muscle, and their primary function is to produce mechanical tension. In mammalian atrial and ventricular tissue, myocytes have roughly a cylindrical shape with dimensions ranging from 50-150 µm in length and 10-20 µm in diameter. The shape and volume of myocytes even in a small region of tissue can be variable and complex, and these properties are also influenced by species, developmental stage, and disease processes.

Myocytes are enclosed by a double phospholipidic membrane, the sarcolemma, which separates the cell exterior from its interior. Cardiac myocytes contain one or more nuclei, usually in the cell centre, mitochondria, myofibrils, the sarcoplasmic reticulum, the sarcomeres and the cytoskeleton, which provides anchoring for the different organelles. The intra-cellular space is filled up with the sarcoplasm, an aqueous solution containing lipids, various ion species, carbon hydrates and proteins. The sarcolemma represents a semi-permeable barrier, and contains the ion channel, pump, and exchanger proteins that carry the inward and outward currents that underlie the electrical activation, as well as proteins involved in cell adhesion and signalling. The myofibrils have distinct, repeating microanatomical units, termed sarcomeres, which represent the basic contractile units of the myocyte. Transverse tubules (t-tubules) are deep invaginations of the sarcolemma, and act to communicate electrical and ion Ca^{++} signals to the cell interior, driving the contraction of the muscle. A further specialisation of the cell membrane is found at its ends, where the intercalated disks couple mechanically the cells. Intercalated disks also include gap junction channels, which provide for inter-cellular electrical
coupling and will be described later on.

Although myocytes account for the largest volume fraction of normal myocardium, fibroblasts are more abundant in number. Cardiac fibroblasts play a major role in the maintenance of the extra-cellular matrix, which is a complex network including strands of fibrous proteins like collagen and elastin and provides a framework for cardiac tissue and is the major determinant of passive mechanical properties. Fibroblasts can develop into myofibroblasts. Both cell types serve as mediators of inflammatory responses and are involved in the development of fibrosis in the injured heart.

Tissue organization

Myocytes are coupled to other myocytes by gap junction channels, which enable both inter-cellular signalling and propagation of the electrical signal. Gap junctions have a cylindrical or barrel shape with a diameter of about 2 nm and length of approximately 2-12 nm and they consist of two hemi-channels, connexons, located in the membrane of the two coupled cells. Each connexon is formed by six membrane proteins, called connexins.

Gap junctions can be found at various locations throughout the sarcolemma, but most are located at intercalated disks. The distribution of gap junctions in the sarcolemma depends on tissue type. In ventricular myocardium, longitudinal gap junctions are most abundant, resulting in macroscopic anisotropic electrical coupling.

The main features of tissue micro-structure are preferential local alignment of myocytes along their principal axis and their end-to-end coupling. A local fiber orientation can be defined along the principal axis of myocytes. In the ventricles, fiber orientation has been known to smoothly rotate between endocardium and epicardium (Figure 1.4) [147]. This finding obtained by visual inspection of tissue has been confirmed using various techniques, such as histology [147], optical techniques like confocal microscopy [140] and diffusion tensor MRI [62]. These techniques are able to provide an accurate description of the fibers orientation if performed ex-vivo, while the application to moving hearts is complex, due to the long acquisition times of these imaging techniques. For this reason a patient-specific description of the fibers direction is generally not available. A common approach in the modeling and simulation of electrocardiology problems is to resort to an analytical description of the fibers, defined on simplified geometries of the hearts (often on an ellipsoidal geometry representing the left ventricle). The analytical field is defined to mimic the main features of the actual cardiac fibers field. An example of this analytical field for a simplified left ventricle geometry can be found in [33].

Comparative anatomical studies have shown that the general features of fibers organization in the ventricular wall are conserved across species. Some of these studies however have highlighted specific regions of the myocardium where fibers organization is much more variable [40, 108, 142].

In addition to the fibrous structure described above, microscopic imaging studies of cardiac tissue have demonstrated that ventricular myocytes are organised into laminar structures, also called sheets. These sheets are typically 4-6 myocytes thick and are sep-
1 Introduction

Figure 1.4: Schematic representation of the fibers arrangement in the myocardium. Modified from [102].

arated by cleavage planes and layers of connective tissue and present multiple branches and discontinuities. The ventricular wall also accommodates a vascular network, the coronary system, and a detailed examination shows the presence of many blood vessels and other voids within the tissue.

We also mention that the myocardium is a heterogeneous tissue and that regional differences in tissue and cell properties occur mainly between atrial and ventricular tissue, but also across the wall thickness.

Finally we report that recent works are devoted to understand the fibroblasts spatial organization, and their possible influence on cardiac electrophysiology. Their structural arrangement in cardiac tissue is still under investigation, even though their organization seems to follow the myocyte organization [64].

Electrical activation

The properties of the cell membrane allow for the existence of an imbalanced ionic charge between the intra-cellular and the extra-cellular spaces and therefore the presence of a potential difference across the membrane, which at rest measures between 80 to 90 mV. The ion species mainly responsible for the variation in time of the transmembrane potential are sodium (\(\text{Na}^+\)), potassium (\(\text{K}^+\)) and calcium (\(\text{Ca}^{++}\)), whose concentration changes under various conditions and stimuli. When a cell is electrically stimulated and the transmembrane potential increases enough, membrane becomes more permeable to specific ions including sodium and potassium. When the transmembrane potential is over a threshold, the inward sodium current dominates over the outward
1 Introduction

potassium one, resulting in a fast depolarization of the membrane, characterized by a reversion of the transmembrane potential which becomes positive. After an overshooting phase, outward potassium current leads to a fast decrease in the transmembrane potential, which returns to the equilibrium state. This process is called repolarization phase. The total potential difference between the pick of depolarization and the resting state is defined action potential, and this fast variation is the signal for the myocardium to contract. Ion concentrations are returned to the rest state by ionic pumps.

The shape of the time-pattern of the transmembrane potential depends on the considered region of the myocardium. Figure 1.5 shows the typical shape of action potential for various type of cardiac cells. Ventricle cells present a plateau phase between the depolarization and the repolarization phase. During this interval an inward calcium current maintains the potential approximately constant (Figure 1.6) and trigger the contraction. Atrial cells, Purkinje fibers and bundle of His show the approximately same behavior, with different plateau length and repolarization slopes, while the sinoatrial node (SA) and the atrioventricular node (AV), that belong to specialized excitatory tissue, do not present the spike nor the plateau phase.

The depolarization is a threshold phenomenon since the action potential develops only if the transmembrane potential reaches a threshold value, otherwise it returns immediately to the resting state. In case the potential exceeds the threshold, the action potential amplitude is independent of the stimulus intensity. After the excitation, cardiac cells are characterized by a refractory period, defined as the amount of time after which a second excitation can occur. This property is important to guarantee that the contraction of the

Figure 1.5: Typical time pattern of action potential in various types of cardiac cells. After http://rezidentiat.3x.ro/eng/tulbritmeng.htm.
1 Introduction

Figure 1.6: Typical time pattern of action potential ventricle cells, with a detailed description of ion currents responsible of each phase. Numbers identify different action potential phases: 0 stands for depolarization, 1 for spike, 2 for plateau, 3 for repolarization and 4 for rest. After http://rezidentiat.3x.ro/eng/tulbritmeng.htm.

cardiac muscle arises only after the previous blood propulsion has ended, maximizing the cardiac output.

In physiological conditions the electrical impulse is generated spontaneously by the specialized muscle cells of the sinoatrial node that lies in the right atrium at the level of its junction with the superior vena cava. This so-called pacemaker has a frequency of about 60-70 beats per minute. The stimulus first spreads through the atria, originating a contraction in the atrial musculature, through which it is transmitted to the atrioventricular node.

Here a brief delay in the conduction occurs, to ensure that the ventricles are stimulated only after the atria have contracted, to allow the ventricles to be completely filled before their contraction starts.

From there, the stimulus reaches the bundle of His, which runs through the fibrous cardiac skeleton, and transmits it to the ventricular myocardium. There it runs first through the bundle branches along the interventricular septum towards the apex and is distributed by the Purkinje fibers over the whole ventricular myocardium. In this way, the cardiac musculature is stimulated along the conduction system and contracts rhythmically.

The cardiac frequency, impulse velocity, and force of contraction are influenced by the autonomic nervous system, that allows to adjust the cardiac activity to the body's needs.

The spread of depolarization through cardiac tissue depends on inter-cellular coupling through gap junctions, as well as intra-cellular and extra-cellular conductivities. Both intra-cellular and extra-cellular spaces are filled with conducting fluid and act as volume conductors, with low-resistance gap junctions connecting the intra-cellular spaces of adjacent cells. This structure ensures that the action potential upstroke in one part of the cell membrane results in depolarization of neighbouring regions that is sufficient to open voltage-gated Na⁺ channels in these regions, resulting in a propagating action
1 Introduction

The electrical properties of bulk tissue depend on the conductivities of the intra-cellular and extra-cellular spaces. In general, both conductivities are anisotropic, and depend on the relative volumes of myocytes and extra-cellular space, blood flow in the vasculature, myocyte shape, the distribution of gap junctions and their state, and the ionic composition of intra-cellular and extra-cellular spaces [146]. Measuring these conductivities in tissue remains a difficult task, depending also on the variable conditions under which experiments are carried out from one study to another. These difficulties explain the large range of experimentally measured values of intra- and extra-cellular conductivities and their anisotropy ratios [130, 146].

The contraction of the cardiac cells is activated by the electrical stimulus, which provokes the opening of the sarcoplasmatic reticulum channels. Calcium ions are then released in the cell, being able to bind to troponin-C. This process triggers the contraction of the sarcomere. The opposite process takes place during the relaxation phase.

Common pathologies of the electrical conduction system

An abnormal excitation and conduction in the myocardium can provoke different physiological consequences to the heart, depending on whether the electrical abnormality evokes a tachycardia, which will limit the time for cardiac filling between beats; evokes a bradycardia, which is inadequate to support sufficient cardiac output; or decreases the coordination of myocytes contraction, which will reduce stroke volume. We report in this Section few examples of abnormalities in the heart electrophysiology.

Supraventricular tachycardia occurs when the atria are abnormally excited and drive the ventricles at a very rapid rate. Supraventricular tachycardia can be generated by two different mechanisms. First, an atrial region, usually outside the SA node, may become irritable (perhaps because of local interruption in blood flow) and begin to fire rapidly to take over the pacemaker function. Such an abnormal pacemaker region is called an ectopic focus. Alternatively, atrial conduction may become altered so that a single wave of excitation does not die out but continually travels around some abnormal atrial conduction loop. In this case, the continual activity in the conduction loop may drive the atria and atrioventricular node at a very high frequency. This self-sustaining process is called a reentry phenomenon and may develop as a result of abnormal repolarization and altered refractory periods in local areas of the myocardium. Atrial flutter is a special form of tachycardia of atrial origin in which a large reentrant pathway drives the atria at very fast rates (250 to 300 beats/min) and normal refractory periods of atrioventricular nodal tissue are overwhelmed.

Conduction blocks occur at the atrioventricular node and generally represent impaired conduction through this tissue. The abnormalities of this kind range from a slow conduction through the atrioventricular (AV) node, to some atrial impulses blocked at the AV node, to no impulses transmitted through the AV node. In case of complete block, some area in the ventricles (often in the common bundle or bundle branches near the exit of the AV node) assumes the pacemaker role for the ventricular tissue. In this case atrial rate and ventricular rate are completely independent, and ventricular rate is likely...
to be slower than normal, even impairing cardiac output.

Atrial fibrillation is characterized by a complete loss of the normally close synchrony of the excitation and resting phases between individual atrial cells. Cells in different areas of the atria depolarize, repolarize, and are excited again randomly. The ventricular rate is often very irregular in atrial fibrillation because impulses enter the atrioventricular node from the atria at unpredictable times. Fibrillation is a self-sustaining process. The mechanisms behind it are not well understood, but impulses are thought to progress repeatedly around irregular conduction pathways sometimes involving a reentry phenomenon. Since atrial conduction plays a minor role in ventricular filling, atrial fibrillation may be well tolerated by most patients as long as ventricular rate is sufficient to maintain the cardiac output. However this disorganized atrial activity may facilitate the formation of blood clots.

Conduction blocks called bundle branch blocks or hemiblocks can occur in either of the branches of the Purkinje system of the intraventricular septum, often as a result of a myocardial infarction. Ventricular depolarization is less synchronous than normal in the half of the heart with the nonfunctional Purkinje system.

Premature ventricular contractions are caused by action potentials initiated by and propagated away from an ectopic focus in the ventricle. As a result, the ventricle depolarizes and contracts before it normally would. A premature ventricular contraction is often followed by a missed beat (called a compensatory pause) because the ventricular cells are still refractory when the next normal impulse emerges from the sinoatrial node. The volume of blood ejected by the premature beat itself is smaller than normal, whereas the stroke volume of the beat following the compensatory pause is larger than normal.

Ventricular tachycardia occurs when the ventricles are paced at high rates, usually by impulses originating from a ventricular ectopic focus. Ventricular tachycardia is a very serious condition since the diastolic filling cannot be complete and the abnormal excitation pathways make ventricular contraction less synchronous and therefore less effective than normal. In addition, ventricular tachycardia often precedes ventricular fibrillation.

Prolonged QT interval, whose name refer to a particular pattern detected by an electrocardiogram, is a result of delayed ventricular myocyte repolarization, which may be due to inappropriate opening of sodium channels or prolonged closure of potassium channels during the action potential plateau phase. Although the normal QT interval varies with heart rate, it is normally less than 40% of the cardiac cycle length (except at very high heart rates). Long QT syndrome (in which the interval is greater than 50% of the cycle duration) may be congenital in origin, may arise from several electrolyte disturbances or may be induced by several pharmacological agents (including some antiarrhythmic drugs). The prolongation of the myocyte refractory period, which accompanies the long QT syndrome, extends the vulnerable period during which extra stimuli can evoke tachycardia or fibrillation. This differs from the ordinary ventricular tachycardia in that the ventricular electrical complexes cyclically vary in amplitude around the baseline and can deteriorate rapidly into ventricular fibrillation.

As atrial fibrillation, ventricular fibrillation, occurs when various areas of the ventricle
are excited and contract asynchronously. In this condition, since some myocardial cells are repolarized and some are still refractory, circus pathways can be triggered easily at this time. Since no pumping action occurs with ventricular fibrillation, the situation is fatal unless quickly corrected by a medical procedure called cardiac conversion. During conversion, the artificial application of large currents to the entire heart, may be effective in depolarizing all heart cells simultaneously and thus allowing to re-establish the normal excitation pathway.

We finally mention myocardial infarction, which is the interruption of blood supply to part of the heart, causing some heart cells to die, with consequent modification of the conduction properties of the tissue. This is most commonly due to occlusion of a coronary artery following the rupture of a vulnerable atherosclerotic plaque. The resulting ischemia (restriction in blood supply) and oxygen shortage, if left untreated for a sufficient period of time, can cause damage or death (infarction) of heart muscle tissue.
2 Some mathematical models for cardiovascular problems

In this Chapter we present the mathematical models considered in the present thesis. In particular, in Section 2.1 we describe models for the heart, including cell-level and tissue-level models for the electrophysiology and mechanical models for the myocardium. We introduce in particular the so-called Bidomain model and its simplified counterpart called Monodomain model. In Section 2.2 we address mathematical models used to describe blood flow in vessels. More precisely, we introduce the Navier-Stokes equations for fluid-dynamics and their generalization to compressible fluids. We also introduce a popular mechanical model suitable for the description of the vessel wall behavior.

2.1 Models for the heart

As we have shown in the Section 1.1.2, the heart activity involves processes which arise at various levels of tissue organisation, from sub-cellular and cellular to the whole organ level, and detailed models for the whole organ come from a synthesis of these different scales.

To develop an integrative description of the whole left ventricle activity, a model for the electrophysiology of a single cardiac cell should be coupled with a model that describes the propagation of the action potential through the cardiac tissue. The resulting coupled model should be integrated with a detailed model of the ventricular anatomy, including both the shape of the organ and the orientation of the cardiac fibers. A mechanical model for the contraction-relaxation of the cardiac wall needs also to be taken into account. To obtain a realistic model of the whole heart, these models should then be coupled with analogous models for the right ventricle and for the atria. An useful introduction to models for the whole ventricle can be found in [30].

In this work we focus on the electrophysiology of the left ventricle, hence we will not address models for the atria (see [35, 110]). We will also provide a brief description of the most common mechanical models available in literature for this purpose, even if we will not perform numerical simulations of the mechanical contraction of the ventricle.

2.1.1 Cell-level models

Cell-level models are intended to describe the time pattern of the transmembrane potential across a single myocyte membrane. Many popular cell-level models are based on
the assumption that the myocyte membrane and its surroundings can be modeled as an electric circuit (Figure 2.1).

![Electric circuit representing a cell membrane.](image)

Figure 2.1: Electric circuit representing a cell membrane. It is composed of a capacitor, with capacitance $C_m$, which works in parallel to ionic channels, each of them represented by a series of a battery and a resistor of constant $g_*$ ($*$ stands for the considered ion). With ECS we denote the extracellular domain and with ICS the intracellular one. $I_m$ is the current flowing through the membrane, $I_c$ is the capacitor current and $u$ is the potential across the membrane. The sum of all the ion channels current is denoted with $I_{ion}$.

In this analogy, the cell membrane is represented by a capacitor, since it is a thin structure composed of a double phospholipidic layer that separates the extra-cellular space (ECS) from the intra-cellular space (ICS). Both ICS and ECS contain various ion species and their motion through ion channels is controlled by the cell membrane which is ion selective. Passage through ion channels, can be governed by a gate, which may be opened or closed by chemical or electrical signals, temperature, or mechanical forces, depending on the variety of channel. These channels in the membrane can be modeled as resistors, whose conductance is time and voltage dependent, and their Nernst potential can be represented by batteries on each channel.

In the sequel we denote with $u$ the transmembrane potential which is defined as the difference between the intra-cellular electric potential and the extra-cellular one:

$$u = u_i - u_e$$

According to the electric circuit representation, the total membrane current $I_m$ is given by the sum of the ion currents flowing through each channel $I_{ion}$ and the capacitor current $I_c = C_m \frac{du}{dt}$, $C_m$ being the membrane capacitance per unit area:

$$I_m = C_m \frac{du}{dt} + I_{ion}$$

(2.1)
To guarantee the conservation of the current, the membrane current must be equal to the current applied to the cell.

The ionic currents can be subdivided into three main groups: inward currents, which depolarise the membrane, increasing the transmembrane potential, outward currents, which repolarise the membrane and currents flowing through non-selective channels, whose direction of flow depends on the membrane voltage.

The most important inward currents are $Na^+$ and $Ca^{++}$ currents, and the most important outward currents are in the $K^+$ current group, which is composed of more than 10 individual currents, each of which has its own features and time dynamics. The ionic currents are in general dependent on the transmembrane potential, ion concentrations and time. Among the ion concentrations, a very important regulator is the intra-cellular $Ca^{++}$ concentration. Hence, modeling the storage, release and uptake of $Ca^{++}$ could be relevant for cardiac cell models, even if it is a complicated task [105].

Currently there are more than one hundred models for cardiac cells available in the literature. They differ both in complexity and in the level of detail with whom they represent the underlying biology. A list of those models can be found in the CellML repository, whose goal is to collect all the validated cell-level models available in the literature.

We will give here a brief introduction on the main groups of models, classified on the basis of their complexity, and focusing on the cell-level models employed in this thesis.

**Phenomenological models**

This class includes the most basic models for the transmembrane potential in a single cell. They are derived through a simplification of the Hodgkin-Huxley model, which is the first mathematical model developed to describe the ion currents through the cell membrane [73]. The Hodgkin-Huxley model, based on experiments on squid giant axons, describes the current of each ion according to Ohm law, taking into account the conductance and the Nernst potential of each ion. The conductance is generally a function of time and transmembrane potential and it is modeled by describing the probability of each channel to be open. The ionic currents considered in this model are the sodium current, the potassium current and a leakage current.

The first and most famous simplified version of the Hodgkin-Huxley model is the FitzHugh-Nagumo model, proposed in [51] for a generic excitable medium. It is composed of a system of two ordinary differential equations whose unknowns are the transmembrane potential $u$ and a recovery variable $w$, which does not have a specific physical meaning but takes into account the refractoriness of the cell after a stimulus.

Variants of the FitzHugh-Nagumo model exist in the literature (see [169]) and some of them adapt its behavior to model cardiac cells (see for instance [4, 128]).

---

The general form of this class of models is
\[
\begin{align*}
\frac{du}{dt} &= f(u, w) \\
\frac{dw}{dt} &= g(u, w),
\end{align*}
\]  
(2.2)

where \(f(u, w)\) is a cubic function of \(u\) and a linear function of \(w\), while \(g(u, w)\) is a linear function of both \(u\) and \(w\).

In particular we detail the Rogers-McCulloch model [128], which reads:
\[
\begin{align*}
C_m \frac{du}{dt} &= -I_{\text{ion}} = -Gu \left(1 - \frac{u}{v_{th}}\right) \left(1 - \frac{u}{v_p}\right) - \eta_1 uw \\
\frac{dw}{dt} &= \eta_2 \left(\frac{u}{v_p} - \eta_3 w\right)
\end{align*}
\]  
(2.3)

In this model \(v_p, v_{th}, G, \eta_1, \eta_2, \eta_3\) and \(C_m\) are parameters related to the membrane properties and define the curve of the action potential. The cubic term is responsible for the activation of the action potential. The analysis of this system in the phase plane, carried out in [128], shows that if the transmembrane potential is over the excitation threshold \(v_{th}\), then it tends to the equilibrium \(u = v_p\), otherwise, the potential tends to the equilibrium \(u = 0\). The main advantage of the Rogers-McCulloch model is that the predicted transmembrane potential does not have the non-physiological undershooting typical of the FitzHugh-Nagumo solution.

The main advantage of the phenomenological models is that they require limited computational resources to compute their numerical solution. Moreover, they capture the most relevant features of the cardiac action potential and the parameters can be tuned to fit the resulting potential to experimental data, when available. Therefore this kind of models can be useful for patient-specific simulations, when various models need to be coupled together and tuned to fit the specific case, without necessarily reproducing all the biophysical details [143]. On the other hand, since the ion channels are not modeled explicitly, it is not possible to simulate the consequences of a pathology involving a specific ion or to fit the time pattern of ion concentrations to available data. To circumvent these disadvantages, biophysically detailed models can be employed.

Biophysically detailed models: first generation

The first generation of cardiac cell models aims at reproducing the action potential based on available experimental information about the ion channels conductance, depending on voltage and time. A review on this kind of models can be found in [134].

The ion channel kinetics adopted in the models of this class is based on the Hodgkin-Huxley description [73], whose general form for channel \(k\) is
\[
I_k(u, w, c) = g_k(c) \prod_{j=1}^{M} w_j^{p_{kJ}} (u - E_k(c)).
\]  
(2.4)
In (2.4) $I_k$ is the transmembrane current across channel $k$, $w$ is the vector of $M$ gating variables, that represent the probability of a single gate of channel $k$ to be open, the exponents $p_{jk}$ link the probability of each gate to be open to the probability of the whole channel $k$ to be open, $c$ is the vector of ion concentrations, $g_k$ is the maximal conductance of channel $k$ and $E_k$ is the Nernst potential for the considered ion. Each gating variable is required to satisfy an ordinary differential equation

$$\frac{dw_j}{dt} = f(w_j)$$  \hspace{1cm} (2.5)$$

with $f(w_j)$ a linear function. The total ion current will be

$$I_{ion} = \sum_{k=1}^{n} I_k,$$

where $n$ is the number of ionic currents. In models of this class, only a few ion concentrations are considered, typically sodium, calcium and potassium, and a subset of them (usually the intra-cellular calcium concentration) is modeled to satisfy a linear ODE, while the remaining ones are neglected.

The very first ionic model for ventricular myocytes is the Beeler-Reuter model [10] and it belongs to this category. It includes the description of four currents (sodium, calcium and two potassium currents) and the intra-cellular calcium concentration is modeled by a linear ODE.

The Luo-Rudy I model, proposed in [91], is an extension of the Beeler-Reuter model. It includes in addition a background current $I_b$ and a plateau current $I_{Kp}$ depending on the transmembrane potential and carried by the potassium ions and a more detailed description of the $K^+$ current, depending on time and on the transmembrane potential. As in the Beeler-Reuter model, the time derivative of the intra-cellular calcium concentration depends on the slow inward current $I_{si}$. The Luo-Rudy I model reads

$$\begin{align*}
C_m \frac{du}{dt} + I_{ion} &= I_{app} \\
I_{ion} &= I_{K1} + I_{Kp} + I_b + I_K + I_{Na} + I_{si} \\
I_{K1} &= f_{K1}(\lbrack K_e \rbrack, u)(u - E_K) \\
I_{Kp} &= f_{Kp}(u)(u - E_K) \\
I_b &= b_1(u + b_2) \\
I_K &= T_K(\lbrack K_e \rbrack) w_3 w_7 (u - E_K) \\
I_{Na} &= g_{Na} w_3^3 w_1 w_2 (u - E_{Na}) \\
I_{si} &= g_{si} w_5 (u - E_{si}) \\
d[Cai]_i dt &= -c_3 I_{si} + c_2 (c_3 - [Ca]_i) \\
\frac{dw_j}{dt} &= \alpha w_j (1 - w_j) - \beta w_j w_j, \quad j \in \{1...6\} 
\end{align*}$$  \hspace{1cm} (2.6)
where $f_{K_1}$ and $f_{K_p}$ are ratios of exponential functions of the transmembrane potential, $I_K$ is a function of the extra-cellular potassium concentration $[K]_e$ and $E_k$ is the Nernst potential of ion $k$; $[Ca]_i$ is the intra-cellular calcium concentration while $g_{Na}, g_{sl}, \alpha_s, \beta_s, b_1, b_2, c_2$ and $c_3$ are parameters determined by fitting to experimental data and $w$ is the vector of gating variables. A complete description of this model is reported in Section A.2.2 or can be found in the original paper [91].

This model has been widely used to simulate the action potential, since it can be easily tuned to reproduce important properties like the Action Potential Duration (APD) and the APD restitution. Moreover, it represents a good compromise between the accurate description of each membrane channel and the easiness of computation of the numerical solution.

Recently a new model for human ventricular cells has been proposed in [154], featuring the same level of complexity as the first generation of biophysically detailed models.

Biophysically detailed models: second generation

This class of models adds, with respect to the first generation models, a detailed description of intra-cellular sodium, calcium and potassium ion concentrations together with the description of additional ion channels. An ODE for each ionic species is added to the model, to guarantee the balance of its concentration, which depends in general on $m$ different types of currents carrying it. The form of this additional ODE for ionic species $S$ reads:

$$
\frac{d[S]}{dt} = \sum_{i=1}^{m} I_S^i(u, [S]),
$$

where $I_S^i$ denotes the $i^{th}$ current carrying the species $S$. Second generation models may also describe the region close to the intra-cellular and extra-cellular surfaces of the membrane.

Examples of this category are the Luo-Rudy II model [90] (an upgraded version of Luo-Rudy I) for guinea pig ventricular cells, DiFrancesco-Noble model for Purkinje cells [41], Winslow model for canine ventricular cells [170] and TNNP model proposed in [153] for human ventricular cells.

Provided that experimental measures are available for specific ionic concentrations and currents, this kind of models can be arbitrarily complicated, adding more and more details. They represent at this time the most accurate models available in the literature for cell membrane behavior. The main drawback is that their numerical solution is computationally expensive and that several experimental measures are required to validate all the model parameters.

Reduced models

Biophysically detailed models yield generally non-linear and stiff systems of equations, whose numerical solution often requires high computational costs, especially when considering the coupling with higher scale models. On the other hand, phenomenological
models can reproduce realistic features, but they are not able to provide any insight into the behavior of specific ion channels. Moreover both the phenomenological and the biophysically detailed models may fail in reproducing, even qualitatively, important properties of the action potential. To circumvent these problems, a new kind of models has been proposed recently. They are derived from biophysically detailed models, by simplifying the underlying electrophysiology and describing only the most relevant currents. An example of this type of models is presented in [47], where the total ion current $I_{ion}$ is considered as the sum of a fast inward sodium current, a slow inward calcium current and a slow outward potassium current. The activation and inactivation of these currents depend on three variables, the transmembrane potential and two gate variables, responsible for inactivation of each current after depolarization and its reactivation after repolarization. The parameters of the models are not directly linked to experimental data, but can be tuned to reproduce the action potential shape and rate dependence typical of the considered cell type. The advantage of this model is that it retains the minimal ionic complexity necessary to reproduce quantitatively the APD restitution curves that describe how the pulse duration (APD) and its propagation velocity (CV) depend on the time interval after repolarization during which the membrane recovers its resting properties.

Other examples of models with this level of details are proposed in [23] and [29].

In this thesis we will model the the action potential at cell-level scale using either the Rogers-McCulloch or the Luo-Rudy I model. However the numerical methods proposed in Chapter 3 can be applied regardless of the cell-level model chosen. To this respect, a possible extension of this work is the implementation of models from the class of second generation biophysically detailed models, or the class of reduced models, by developing an interface for exporting CellML repository models to our electrocardiology solver in LifeV software library (see next Chapter).

2.1.2 Tissue-level models

At cellular scale the propagation of the action potential from one cell to its adjacent cells is a discrete process. However, at larger spatial scales depolarisation appears to propagate smoothly as described in [43]. Therefore it is common in the literature to assume that the discrete nature of potential propagation can be neglected at tissue scale, considering the cardiac tissue as a functional syncytium. In this work we employ a mathematical formulation of the electrocardiology problem based on this assumption. As reported in [28], there exist in the literature some examples of approaches that do not neglect the discrete nature of the action potential propagation. As examples we mention a model based on cellular automata and coupled map lattices, presented in [74], and lattices of coupled ordinary differential equations, described in [171].
The Bidomain model

A continuous model for the action potential propagation in the myocardium can be derived using a homogenization technique in a rigorous mathematical framework, as formalized in [34, 84].

We will report here the most common derivation of the continuous model, adopted e.g. in [145], based on the volume averaging technique [167]. With this approach the values of parameters and variables in each point of the domain are obtained by taking the average of the considered quantity on a volume centred in the considered point. The dimension of the region on which to compute the average should be large enough to contain a discrete amount of cells but small with respect to the scale of the considered tissue.

Volume averaging techniques are applied as done when studying porous media. In our case the intra-cellular potential is defined only in the intra-cellular space, while the extra-cellular potential is defined only on the extra-cellular space, therefore the model includes variables or parameters defined only on part of the domain.

Thanks to this averaging procedure, we can assume that both the intra- and extra-cellular domains are coexisting in each point of the cardiac domain \( \Omega \subset \mathbb{R}^3 \), but separated by the cell membrane. All the quantities obtained by averaging on the extra-cellular domain will be denoted by the subscript \( e \) while those obtained by averaging on the intra-cellular domain will be denoted by the subscript \( i \).

Since it takes into account both the intra-cellular and the extra-cellular domains, this model is called Bidomain and the procedure to derive it is based on balance of current flows.

We first introduce the tensors \( D_i \) and \( D_e \), representing the conductivity tensors of the intra-cellular and extra-cellular medium, respectively. These tensors quantify the ability of the action potential to propagate in the two media. They depend on the position in the domain and on the local orientation of the cardiac fibers and describe the conductivity in the tissue. To define them it can be useful to set up a local reference system \( \{a_l, a_t, a_n\} \) in each point, based on the anatomical structure of the tissue, where \( a_l \) is a longitudinal unit vector, parallel to the local fiber, \( a_t \) is a transversal unit vector, orthogonal to the local fiber and laying in the sheet plane and \( a_n \) is a normal unit vector orthogonal to the previous ones (see Figure 2.2). We may then introduce the conductivity values of each medium, which can also differ according to the considered direction and are denoted by \( \sigma_{l,e}^{i} \) in \( a_l \) direction, \( \sigma_{t,e}^{i} \) in \( a_t \) direction and \( \sigma_{n,e}^{i} \) in \( a_n \) direction.

The conductivity tensors can therefore be expressed as

\[
D_{i,e}(x) = \sigma_{l,e}^{i} a_l a_l^T + \sigma_{t,e}^{i} a_t a_t^T + \sigma_{n,e}^{i} a_n a_n^T.
\] (2.7)

For easiness of notation we have omitted the dependence of \( \sigma_{l,t,n}^{i,e} \) and \( a_{l,t,n} \) on the position \( x \), but we consider those quantities as non-uniform in the domain.

If the tissue is assumed to be axial isotropic, then within each medium the transversal and normal conductivity values are equal and the expression of \( D_{i,e} \) can be simplified:

\[
D_{i,e}(x) = \sigma_{l,e}^{i} I + (\sigma_{l,e}^{i} - \sigma_{i,e}^{i}) a_l a_l^T, \quad \sigma_{l,e}^{i} = \sigma_{i,e}^{i}.
\] (2.8)
2 Some mathematical models for cardiovascular problems

Figure 2.2: Anatomical structure of the tissue. In cyan we display the longitudinal direction $\mathbf{a}^l$, in green the transversal direction $\mathbf{a}^t$ and in red the normal direction $\mathbf{a}^n$. Modified from [102].

Usually it is assumed that the conductivity tensors fulfill a uniform elliptic condition in $\Omega$.

The description of each domain is based on the Ohm’s law defining the relationship between the electric potentials $u_{i,e}$, the current densities $J_{i,e}$, and the conductivity tensors $D_{i,e}$ as:

$$J_{i,e} = -D_{i,e} \nabla u_{i,e}.$$  \hspace{1cm} (2.9)

If no charge source is present in ICS and ECS, then the net current flux between the intra- and the extra-cellular domain is assumed to be zero as a consequence of the charge conservation in an arbitrary portion of tissue, yielding:

$$\int_{\Gamma_m} J_i \cdot n d\Gamma_m = - \int_{\Gamma_m} J_e \cdot n d\Gamma_m,$$

where $\Gamma_m$ denote the membrane separating ICS from ECS and $n$ is the normal unit vector to the surface $\Gamma_m$ and outgoing from ICS. The current flux outgoing from ICS is also equal to the membrane current $I_m$ flowing through $\Gamma_m$. Exploiting the divergence theorem we get

$$\nabla \cdot D_i \nabla u_i = \chi I_m$$
$$\nabla \cdot D_e \nabla u_e = -\chi I_m$$

Thanks to (2.1), and defining $\chi$ as the ratio between the membrane area and the cell volume, we obtain the Bidomain model in absence of external stimuli:

$$\begin{cases}
\chi \left( C_m \frac{du}{dt} + I_{ion}(u, w, c) \right) = \nabla \cdot (D_i \nabla u_i) \\
\chi \left( C_m \frac{du}{dt} + I_{ion}(u, w, c) \right) = -\nabla \cdot (D_e \nabla u_e)
\end{cases}$$  \hspace{1cm} (2.10)
This model needs to be completed by proper boundary and initial conditions. Generally, the myocardium is assumed to be isolated from the surrounding tissue, corresponding to homogeneous boundary conditions

\[
\begin{align*}
    n^T D_i \nabla u_i &= 0 \\
    n^T D_e \nabla u_e &= 0.
\end{align*}
\]  

(2.11)

We will assume (2.11) to hold in the sequel, unless otherwise stated. The initial condition can be expressed as

\[u(x, 0) = u_0(x),\]

where \(u_0 : \Omega \rightarrow \mathbb{R}\) is generally a constant function on the domain, whose value depends on the ionic model chosen.

If current stimuli are applied to the ICS or ECS, the balance expressed in (2.10) should be modified accordingly, introducing \(I_{\text{app}} : \Omega \times (0, T) \rightarrow \mathbb{R}\) in the formulation. The complete Bidomain formulation is finally

\[
\begin{align*}
    \chi C_m \frac{\partial u_i}{\partial t} - \nabla \cdot (D_i \nabla u_i) + \chi I_{\text{ion}}(u, w) &= I_{i, \text{app}} \quad \text{in } \Omega \times (0, T) \\
    - \chi C_m \frac{\partial u_e}{\partial t} - \nabla \cdot (D_e \nabla u_e) - \chi I_{\text{ion}}(u, w) &= -I_{e, \text{app}} \quad \text{in } \Omega \times (0, T) \\
    u(x, t = 0) &= u_i(x, t = 0) - u_e(x, t = 0) = u_0 \quad \text{in } \Omega \\
    n^T D_i \nabla u_i &= 0 \quad \text{on } \partial \Omega \times (0, T) \\
    n^T D_e \nabla u_e &= 0 \quad \text{on } \partial \Omega \times (0, T),
\end{align*}
\]  

(2.12)

where \(n\) is the unit vector normal to the tissue surface and pointing outward.

System (2.12) is referred to as symmetric formulation of the Bidomain and needs to be coupled with a cell-level model for \(I_{\text{ion}}\), as described in Section 2.1.1. If homogeneous Neumann boundary conditions are prescribed, the intra-cellular and extra-cellular applied current stimuli must satisfy a compatibility condition, which reads:

\[
\int_{\Omega} I_{i, \text{app}} d\mathbf{x} = \int_{\Omega} I_{e, \text{app}} d\mathbf{x}.
\]

The Bidomain system (2.12) is composed of two parabolic reaction diffusion equations in the intra- and extra-cellular potential \(u_i\) and \(u_e\). By re-writing the Bidomain system in matrix form

\[
\chi C_m \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \frac{\partial}{\partial t} \begin{bmatrix} u_i \\ u_e \end{bmatrix} - \begin{bmatrix} \nabla \cdot D_i \nabla u_i \\ \nabla \cdot D_e \nabla u_e \end{bmatrix} + \chi \begin{bmatrix} I_{\text{ion}}(u, w) \\ -I_{\text{ion}}(u, w) \end{bmatrix} = \begin{bmatrix} I_{i, \text{app}} \\ -I_{e, \text{app}} \end{bmatrix}
\]  

(2.13)

it can be noticed that a singular matrix multiplies the time derivative of the unknown vector \([u_i, u_e]^T\) and the Bidomain system is therefore defined degenerate.

The transmembrane potential \(u\) is uniquely determined, while \(u_i\) and \(u_e\) are determined up to the same function of time. The uniqueness is usually recovered by imposing that \(u_e\) has zero average on \(\Omega\).
Let us define $V = H^1(\Omega) \times H^1(\Omega) \backslash \{(c, c) : c \in \mathbb{R}\}$ and denote by $(\cdot, \cdot)$ the scalar product in $L^2$. The variational form of the Bidomain problem reads as follows: given $I_{\text{app}}^i$ and $I_{\text{ion}}$, find $[u_i, u_e] \in V$ such that

$$\chi C_m (\frac{\partial u}{\partial t}, \phi) + a_i(u_i, \phi_i) + a_e(u_e, \phi_e) + (I_{\text{ion}}(u), \phi) = (I_{\text{app}}^i, \phi_i) - (I_{\text{app}}^e, \phi_e) \quad (2.14)$$

for each $[\phi_i, \phi_e] \in V$, where $\phi = \phi_i - \phi_e$. The forms $a_\tau(v, \phi)$ are defined as $a_\tau(v, \phi) = \int_\Omega \nabla v^T D_\tau \nabla \phi \, dx$ (for $\tau = i, e$).

We introduce the following notation, since it will be used in Section 3.3. We define the norm $\|u_i, u_e\|_{E(\Omega)}^2$ of the solution as

$$\|u_i, u_e\|_{E(\Omega)}^2 = \int_\Omega (\nabla u_i)^T D_i \nabla u_i + \int_\Omega (\nabla u_e)^T D_e \nabla u_e, \quad (2.15)$$

which is a semi-norm in $V$ since the two tensors $D_i$ and $D_e$ are assumed to be positive.

The well-posedness of the Bidomain problem coupled with the FitzHugh-Nagumo ionic model has been proven in [34], while for Luo-Rudy I and more general ionic models we refer to [161].

We mention that the Bidomain model can in principle be extended to include a more detailed description of the tissue. Recently a further domain representing fibroblasts has been added to the standard Bidomain, obtaining the model presented in [137].

The Monodomain model

The degenerate nature of the Bidomain system causes the matrix corresponding to its fully discrete approximation to be ill-conditioned. As a consequence, the numerical solution of the Bidomain problem implies high computational costs.

Due to this observation, to simulate the action potential propagation in the myocardium a common approach in the literature is to turn to a simplified version of the Bidomain, called Monodomain problem.

It can be derived from (2.12) by assuming that the intra-cellular and extra-cellular conductivity tensors have the same anisotropy ratio and hence they are proportional by a constant $\lambda \in \mathbb{R}$. This simplifying assumption reads

$$D_e = \lambda D_i. \quad (2.16)$$

Under assumption (2.8), the parameter $\lambda$ can be chosen, for instance, by minimizing the functional

$$J = (\sigma_e^i - \lambda \sigma_i^i)^2 + 2 (\sigma_e^i - \lambda \sigma_i^i)^2$$

for given values of the conductivities. Another way of selecting $\lambda$ has been proposed in [107]. In general, if we define

$$\lambda_m = \min \left\{ \frac{\sigma_e^i}{\sigma_i^i}, \frac{\sigma_e^i}{\sigma_i^i} \right\}, \quad \lambda_M = \max \left\{ \frac{\sigma_e^i}{\sigma_i^i}, \frac{\sigma_e^i}{\sigma_i^i} \right\}, \quad (2.17)$$

29
it is reasonable to choose $\lambda_m \leq \lambda \leq \lambda_M$.

By taking a linear combination of the equations in (2.10), with coefficients $\frac{\lambda}{1+\lambda}$ and $-\frac{1}{1+\lambda}$, we get the Monodomain partial differential equation:

$$
\chi C_m \frac{\partial u}{\partial t} - \nabla \cdot (D^M \nabla u) + \chi I_{ion} = I_{app},
$$

(2.18)

where $D^M = \frac{\lambda D_i}{1+\lambda}$ and $I_{app} = \frac{I_{app}^i + I_{app}^e}{1+\lambda}$. As in the Bidomain case, it must be completed by initial and boundary conditions that stem from the Bidomain problem under the simplifying assumption (2.16). The complete Monodomain problem is

$$
\begin{cases}
\chi C_m \frac{\partial u}{\partial t} - \nabla \cdot (D^M \nabla u) + \chi I_{ion} = I_{app} & \text{in } \Omega \times (0,T) \\
u(x, t = 0) = u_0 & \text{in } \Omega \\
n^T D^M \nabla u = 0 & \text{on } \partial \Omega \times (0,T),
\end{cases}
$$

(2.19)

to be coupled again with a suitable cell-level model.

The advantage of this formulation is the reduced computational effort required for its numerical solution, as reported in [30]. Actually the Monodomain problem involves only the transmembrane potential, leading to an important reduction of the problem size, and to the much better conditioning of this simplified problem.

We point out that in [84] a different procedure to derive the Monodomain equation is presented, which proposes to mediate the contribution of the intra- and extra-cellular conductivities in the definition of the Monodomain conductivity tensor $D^M$. The resulting equation is the same as in (2.19) but the conductivity tensor is defined as

$$
D^M = D_e(D_i + D_e)^{-1} D_i.
$$

In this work we assume (2.16) to hold when dealing with the Monodomain approximation.

The differences between the Bidomain and Monodomain solution is particularly relevant when an injection of current in the extra-cellular space is included in the model, as shown in [61], and the unequal anisotropy of the intra-cellular and extra-cellular spaces plays an important role in the simulation of a defibrillation [168]. In [139] Bidomain simulations show a different pattern of reentrant waves close to the tissue surface, with respect to the results of Monodomain simulations. Another situation where the hypothesis of proportionality between the conductivity tensors could prevent the simulation of a realistic pattern is a pathological conditions where alterations of the electric conductivity occur in specific subregions of the domain, affecting differently the ICS and the ECS. Finally, the differences between the two solutions tends to concentrate in the surroundings of the propagating front [72] and the front propagation predicted by the considered version of the Monodomain is slower than the one predicted by the Bidomain model [123].
Formulation in moving domains

We present hereafter a formulation of the Bidomain model that can be used to simulate the action potential propagation in a moving domain, after a quick introduction to the kinematics quantities that are necessary to the presentation, following [53].

Let us denote with \( \Omega \subset \mathbb{R}^3 \) the reference configuration of the considered continuum. \( \hat{\Omega} \) is assumed to be a bounded, open and simply connected subset of \( \mathbb{R}^3 \).

A deformation of \( \hat{\Omega} \) is defined as a smooth one-to-one mapping
\[
\hat{\phi} : \hat{\Omega} \rightarrow \Omega, \quad \hat{x} \rightarrow x = \hat{\phi}(\hat{x}),
\]
that associate each point \( \hat{x} \) of the reference configuration to a position \( x \) in the current one. The difference between the position of a considered point in the current configuration and its position in the reference one is called displacement:
\[
d(\hat{x}) = \hat{\phi}(\hat{x}) - \hat{x}.
\]

It is possible to define a second order tensor \( \hat{\mathbf{F}} : \Omega \rightarrow \mathbb{R}^{3 \times 3} \), named deformation gradient, which represents the gradient of the deformation with respect to the coordinates in the reference configuration.

A motion of a continuum body is defined as a smooth map
\[
\hat{\varphi} : \hat{\Omega} \times \mathbb{R}^+ \rightarrow \mathbb{R}^3, \quad (\hat{x}, t) \rightarrow x = \hat{\varphi}(\hat{x}, t),
\]
such that at any \( t \geq 0 \), \( \hat{\varphi} \big|_t \) is a deformation.

The current configuration at time \( t \) is \( \Omega(t) = \hat{\varphi}(\hat{\Omega}, t) \) and the reference configuration can be assumed to be the initial domain \( \Omega_0 \). The displacement is a function of time:
\[
d(\hat{x}, t) = \hat{\varphi}(\hat{x}, t) - \hat{x}
\]
while the deformation gradient is
\[
\hat{\mathbf{F}}(\hat{x}, t) = \nabla_{\hat{x}} \hat{\varphi}(\hat{x}, t).
\]

The determinant of \( \hat{\mathbf{F}} \) is the Jacobian of the deformation and is denoted by \( \hat{J} \). It is assumed to be strictly positive in the domain, in order for the deformation to have an inverse, and it measures the variation of volume due to the deformation.

Thanks to (2.20), the deformation gradient can be expressed as
\[
\hat{\mathbf{F}} = \mathbf{I} + \nabla_{\hat{x}} d,
\]
where \( \mathbf{I} \) is the 3 by 3 identity matrix.

By definition of \( \hat{\mathbf{F}} \), we have
\[
\hat{\varphi}(x + d, t) - \hat{\varphi}(x, t) = \hat{\mathbf{F}}(\hat{x}, t)[d] + o(d).
\]
This relationship can be exploited to find an approximation of the cardiac fibers field, during the motion, given the correspondent field in the reference configuration. In
particular, assuming (2.8) to hold, only the fiber direction \( \mathbf{a}(t) \) at time \( t \) needs to be computed, and it can be approximated by

\[
\mathbf{a}(t) = \frac{\hat{F} \hat{\mathbf{a}}}{\| \hat{F} \hat{\mathbf{a}} \|},
\]

where \( \hat{\mathbf{a}} \) represent the fibers field in the reference configuration.

We can now formulate the Bidomain equations in the current configuration domain \( \Omega(t) \) as

\[
\begin{align*}
\chi C_m \frac{\partial u}{\partial t} - \nabla \cdot (D_i(t) \nabla u_i) + \chi I_{\text{ion}}(u, w) &= I_{\text{app}}^i \quad \text{in } \Omega(t) \times (0, T) \\
\chi C_m \frac{\partial u}{\partial t} - \nabla \cdot (D_e(t) \nabla u_e) - \chi I_{\text{ion}}(u, w) &= -I_{\text{app}}^e \quad \text{in } \Omega(t) \times (0, T) \\
u(\mathbf{x}, t = 0) &= u_i(\mathbf{x}, t = 0) - u_e(\mathbf{x}, t = 0) = u_0 \quad \text{in } \Omega(t) \\
\mathbf{n}^T(t) D_i \nabla u_i &= 0 \quad \text{on } \partial \Omega(t) \times (0, T) \\
\mathbf{n}^T(t) D_e \nabla u_e &= 0 \quad \text{on } \partial \Omega(t) \times (0, T),
\end{align*}
\]

where we have highlighted the dependence on \( t \) of the conductivity tensors (through the fibers field), of the domain and of the normal unit vector \( \mathbf{n} \), since its direction depends on the domain boundary.

The motion of the tissue is affecting the Bidomain equations at different levels. The conductivity tensors depend actually on the displacement field \( d \), transforming the fibers field on which the tensors are defined. In [103] also the dependence of the membrane capacitance \( C_m \) and the ionic current \( I_{\text{ion}} \) on the displacement field is assumed. The time derivative of the potential should be intended as a Lagrangian one, since it is related to the transmembrane dynamics in a specific tissue element. Moreover, in principle the motion of the domain could induce a modification of the reference system, implying that the space derivatives should be taken over a curvilinear surface. This aspect of the problem will be object of further analysis.

The formulation (2.22) of the Bidomain problem can be employed to perform a numerical simulation of the action potential propagation in a moving domain, provided that the deformation \( \varphi \) (or the displacement vector \( d \)) is known. In this case, the computational domain and the fiber field must be updated during the simulation, as described in Section 4.3.

We point out that other approaches can be used for the same purpose. In particular it is common to re-write on a reference fixed configuration a problem that is naturally set on a moving domain, pulling back the equations to a fixed domain. The formulation in a fixed domain is the best choice whenever the deformation of the domain is not known a priori and need to be computed, and it is generally exploited with solid mechanics

\[
\frac{\partial u}{\partial t} = \lim_{\Delta t \to 0} \frac{u(\mathbf{x}(t + \Delta t), t + \Delta t) - u(\mathbf{x}(t), t)}{\Delta t}
\]
models, as we will see in Sections 2.1.3 and 2.2.2. We direct the readers to [14, 66] for a comprehensive description of this type of formulation and an introduction to the associated mechanics variables.

2.1.3 Myocardium mechanics

Let us introduce some basic definitions of continuum mechanics that will be useful in this Section.

We will denote with $\sigma$ the Cauchy stress tensor, which is a symmetric second order tensor whose action on a vector normal to a given surface represents the Cauchy stress exerted on that surface. The first Piola-Kirchhoff tensor is defined as

$$\hat{\Pi} = \hat{J} \sigma \hat{F}^{-T},$$

and defines the stress tensor in a reference configuration. In continuum mechanics, the properties of a material are defined by a constitutive law, that relates a stress tensor to kinematics quantities. In solid mechanics, in particular, the constitutive law of the considered continuum can be expressed as the relation between the stress tensor $\hat{\Pi}$ and the Green-Lagrange strain tensor, defined as

$$\hat{E} = \frac{1}{2} (\hat{F}^T \hat{F} - I),$$

where $I$ is the 3 by 3 identity matrix. A different stress tensor can be defined, called second Piola-Kirchhoff tensor

$$\hat{\Sigma} = \hat{F}^{-1} \hat{\Pi}$$

whose advantage is that it is symmetric, unlike the first Piola-Kirchhoff tensor.

Very often (see [56]) biological tissues are modeled as hyperelastic material, a subclass of elastic materials. A continuum is said to be elastic if the stress is a function of the current deformation but is independent of the deformation history, while it is defined hyperelastic if there exists a scalar function $\omega$, called deformation energy such that

$$\hat{\Sigma}(\hat{E}) = \frac{\partial \omega}{\partial \hat{E}}(\hat{E}).$$

We also recall the peculiarity of pseudoelastic materials: materials in this class feature a constitutive law with different properties during loading and unloading processes, but substantially independent of the strain rate [56]. For further details on these topics, the interested reader can refer to [14] and [112] among the others.

Let us focus now on the myocardial tissue. From uniaxial, biaxial and (less often) whole heart tests the main mechanical properties of the tissue have been estimated. In particular the stress-strain relationship appears to be non-linear and large deformations take place during the cardiac cycle. A minor hysteretic behavior has been observed, the stress-strain relationship being an exponential function with different parameters values during the loading and the unloading process. The structure of the sarcomeres induces
also the presence of a limit deformation, over which the myocyte is more resistant to the
deformation. The organization of the tissue in fibers makes the mechanical behavior of
the tissue anisotropic. In particular the stiffness in the fiber direction is approximately
double with respect to the one in the transversal direction. An overview on experimental
evidences of myocardium mechanical properties can be found for instance in [63]. In
[57] the presence of residual stresses in the heart, namely the presence of non-zero stress
without external load on the tissue, has been proven experimentally.

In the literature various kind of models can be found that approximate the main fea-
tures of the tissue here described. Generally it is assumed that the cardiac tissue is an
incompressible and hyperelastic tissue (neglecting the presence of the blood flow in the
tissue).

In cardiac mechanics, the passive behavior (during the diastole) is generally modeled
separately from the active behavior (during the systole).

**Passive behavior**

Let us first consider the passive or diastolic phase. Mechanical models for the description
of this behavior can be classifieied as phenomenological models or structural approach
models. Models in the first class specify a function for the deformation energy able to
reproduce the stress-strain measured curves. The second approach consists in obtaining
the deformation energy of the tissue by summing up the deformation energy of the most
important constituents of the tissue. As expected, the advantage of this second approach
is that it is more realistic, while its disadvantage is the more complex form of $\omega$ and its
dependence on a larger number of parameters to be estimated.

The general form of the deformation energy of an incompressible, axially isotropic
hyperelastic material is

$$\omega = F_1(I_1, I_2) + F_2(I_4) + F_3(I_1, I_2, I_4),$$

where $I_1$ and $I_2$ are the first and second invariant of the Cauchy-Green deformation
tensor and $I_4$ is the stretch ratio in the fibers direction (see [166]). Among the phe-
nomenological models, the first deformation energy function was proposed by Fung
in [55] and consists of an exponential term summed to a linear term in the components
of $\dot{E}$. This expression has been modified in [63], Chapter 1, to fit biaxial experimental
data, and in [63], Chapter 5 and [63], Chapter 6, to include directly the incompress-
ability constraint. This kind of models have been compared with biaxial tests results
and simulated, in a finite element framework, on simplified 3D geometries, giving good
accordance with experimental evidences.

The model proposed in [78] is based on the assumption that a non-linear, pseudoelas-
tic, incompressible, locally homogeneous and locally isotropic material can be described
by a deformation energy function depending only on the first invariant of the Cauchy-
Green deformation tensor and on the stretch ratio in the fiber direction. The form pro-
posed, in accordance with experimental observations, is a polynomial function of $I_1$ and
the stretch ratio, whose parameters can be estimated through a linear regression [78].
Some mathematical models for cardiovascular problems

The application of this model has been compared with 1D data in the fiber direction and its orthogonal direction, showing good accordance.

In [102] the authors observe that in the myocardial tissue the stress is very small for small deformations, while it increases dramatically when the deformation approaches its limit. On this basis they propose a “pole-zero” constitutive law, of the form

$$\omega = \sum_{i=1}^{3} \sum_{j=1}^{3} k_{i,j} \frac{E_{i,j} E_{j,i}}{[a_{i,j} - E_{i,j}][b_{i,j}]}$$

where $k_{i,j}$, $a_{i,j}$ and $b_{i,j}$ are parameters to be estimated and $E_{i,j}$ are components of the deformation tensor. The number of parameters to be estimated can be reduced, exploiting anatomical observations. This model has been included in a finite element model of the left ventricle and validated against 3D data concerning the principal deformations of the tissue.

The example of a structural approach model that we report here is introduced in [77] and is based on the consideration that the main elements of the myocardium are fibers and collagen networks in a fluid matrix. The deformation energy of the myocardium is therefore derived by summing the deformation energy of the muscular fibers and of the collagen, weighting them with the occupied volume fraction. To take into account the variability of each family of constituents within the tissue, concerning for example the spatial orientation of the fibers bundle, a stochastic approach is employed, defining the probability of a certain configuration to appear. This model is based on a large number of parameters and their value can be estimated by a least-square fitting of the experimental data on the stress-deformation curve derived by the model. Biaxial tests prove the accuracy of the approach, whose main disadvantage is its complexity.

Active behavior

The most common approach in the literature to model the active contraction of the heart muscle, triggered by the electrical signal, is to modify the stress tensor which describes the myocardium in diastole by adding a term representing the active component. Generally it is assumed that the active tension acts only in the direction longitudinal to the cardiac fibers. This active component $T_a$ can be modeled again by phenomenological models or by structural detailed models. However, since the intracellular phenomena which trigger the contraction are complex, involving chemical reactions and microscopic dynamics, whose description does not fit with the purpose of this introduction, we focus here only on phenomenological models for the active tension. Examples of biophysically detailed models can be found in [65] and [138].

We mention that recently in [100] a different approach has been proposed to model the cardiac activation, applied to cardiac electro-mechanical coupling in [24]. In particular the authors introduced a multiplicative decomposition of the deformation gradient $F$ into an active component $F_a$, measuring the change of length of the tissue due to the muscle contraction, and an elastic component $F_e$. At any given time $t$, the active deformation field $F_a$ describes the active deformation of a muscle fiber. This approach
Some mathematical models for cardiovascular problems

differs from the most common one since a multiplicative decomposition is applied to the
deformation gradient instead of an additive decomposition on the stress tensor. We refer
to the aforementioned papers for the details of this new approach.

A simple phenomenological model for the active tension in a cardiac electro-mechanical
coupling is proposed in [103]. It is based on an ordinary differential equation

$$\frac{dT_a}{dt} = c(u)(k_{Ta}u - T_a)$$

where $c(u)$ is a step function depending on the transmembrane potential value and $k_{Ta}$
controls the entity of the active stress and needs to be estimated.

A more accurate and complex model is proposed originally in [81] and then modified in [80] and [101]. This approach consists in modeling the isometric tension $T_i$, namely the tension that arises in the first phase of the systole without inducing changes
in sarcomeres length, and to relate the active tension $T_a$ to $T_i$ by means of a non-linear
function of the recent history of the time-derivative of the stretch ratio, derived by exper-
imental measures. The isometric tension is modeled as a function of time, of the stretch
ratio in the fiber direction and of the calcium ions concentration in the cell. The intra-
cellular calcium concentration can be modeled in turn as a time dependent function or
through its relationship with other intracellular ions. We refer to the cited bibliography
for the details on the model.

The validation of active behavior models is still a challenging topic, since experimental
measures take automatically into account also the passive behavior. Therefore an
evaluation of the active model alone is not possible.

An interesting technique is employed in [156] to validate a mechanical model for
the cardiac mechanics (see [18]), where the contractile behaviour of the sarcomeres is
described by a Hill’s three-elements model. It is based on Magnetic Resonance Tagging
data, which can provide information on the myocardial strains in a non-invasive way.
This strain field can then be compared with the one computed by the mathematical
model at hand, evaluating its predictive ability. This validation procedure seems to be
reliable and it has been able to give information on the sensitivity of the strains to the
setting of the fiber orientation in the model.

The choice of a model for the description of the cardiac mechanics is non trivial. A
possible criterion is to choose the model that best fits with experimental data. However,
very often the experimental tests are performed on uniaxial or biaxial specimens, and
the three dimensional behavior may be badly approximated. The choice should also
take into account the complexity of the model, since the coupling of such a model with
an ionic model and an electrical propagation model can lead to very high computational
costs, if each component presents high complexity. In summary, the chosen model should
be a trade-off between accuracy and computational effort required by the numerical
simulations.
2 Some mathematical models for cardiovascular problems

2.2 Models for blood fluid dynamics in vessels

In this Section we focus on the mathematical description of the fluid dynamics in large vessels, introducing some mathematical models that will be used in Chapter 4. This sketch is based on [53], Chapter 3.

We first present in Section 2.2.1 the well-known Navier-Stokes equations, describing the velocity and pressure field of an incompressible Newtonian fluid. We will also mention a generalization of this model for a Navier-Poisson fluid, since such model appears in the fluid-registration algorithm presented in Section 4.2.1. In Section 2.2.1 we introduce the Arbitrary Lagrangian Eulerian (ALE) formulation, convenient when dealing with fluid dynamics simulations in moving domains. Finally, we report in Section 2.2.2 a common model used to approximate the behavior of the arterial vessel wall.

2.2.1 Navier-Stokes equations

The partial differential equations that model the dynamics of a generic continuum in a domain $\Omega$ can be derived from mass and momentum conservation principles of continuum mechanics.

Let us consider a continuum with density $\rho$ and velocity $u$. The mass conservation equation reads:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho u) = 0, \quad \text{in } \Omega, \ t > 0. \quad (2.23)$$

If the material density is constant, (2.23) implies

$$\nabla \cdot u = 0, \quad \text{in } \Omega, \ t > 0, \quad (2.24)$$

which is called incompressibility equation.

The momentum conservation can be expressed as

$$\rho \frac{\partial u}{\partial t} + \rho (u \cdot \nabla) u - \nabla \cdot \sigma = \rho f, \quad \text{in } \Omega, \ t > 0, \quad (2.25)$$

where $\sigma$ is the Cauchy stress tensor.

To focus on fluid materials, we need to introduce the strain rate tensor $D$, defined as

$$D = \frac{1}{2} (\nabla u + \nabla u^T).$$

Let us first consider a Newtonian incompressible fluid, with constant density. In this case the constitutive relation reads:

$$\sigma(u, p) = -p I + 2\mu D(u), \quad (2.26)$$

where $p$ is the pressure field, $I$ is the 3 by 3 identity matrix and $\mu$ is the dynamic viscosity of the fluid. A material satisfying this constitutive law is commonly used to model blood flow in large arteries. Nevertheless more accurate rheological models have been developed, useful to take into account non-Newtonian characteristics particularly evident at
Some mathematical models for cardiovascular problems

low shear rate. We will not consider non-Newtonian blood flow in this discussion and we refer e.g. to [53], Chapter 6, for a presentation on this topic.

By plugging (2.26) in (2.25) and considering (2.23) we obtain the Navier-Stokes equations for a constant density fluid:

\[
\begin{cases}
\rho \frac{\partial u}{\partial t} + \rho (u \cdot \nabla) u + \nabla p - 2 \nabla \cdot (\mu D(u)) = \rho f, & \text{in } \Omega, t > 0 \\
\nabla \cdot u = 0, & \text{in } \Omega, t > 0
\end{cases}
\] (2.27)

This system is completed by prescribing an initial condition \( u_0 \) on fluid velocity:

\[ u(t = 0, x) = u_0(x), \quad \text{in } \Omega \]

and proper boundary conditions. Generally a Neumann boundary condition is prescribed on a portion of the boundary \( \Gamma_N \) while prescribing a Dirichlet boundary condition on the remaining part, called \( \Gamma_D \). A possible set of boundary conditions for problem (2.27) is

\[
\begin{align*}
- p n + 2 \mu D(u) n &= h & \text{on } \Gamma_N \subset \partial \Omega \\
u &= g & \text{on } \Gamma_D \subset \partial \Omega
\end{align*}
\] (2.28)

with \( \Gamma_N \cup \Gamma_D = \partial \Omega \) and \( \Gamma_N \cap \Gamma_D = \emptyset \). In (2.28) \( h \) and \( g \) are given functions \( g : \Gamma_D \times \mathbb{R}^+ \rightarrow \mathbb{R}^3, h : \Gamma_N \times \mathbb{R}^+ \rightarrow \mathbb{R}^3 \). In case \( \Gamma_N = \emptyset \) the Dirichlet boundary condition must satisfy the condition \( \int_{\partial \Omega} g \cdot n = 0 \) to ensure compatibility with \( \nabla \cdot u = 0 \).

Let us define the functional space

\[ V = \{ v \in [H^1_{\Gamma_D}(\Omega)]^d : \nabla \cdot v = 0 \}. \]

The weak formulation of the Navier-Stokes equations (2.27) reads: for almost every \( t > 0 \) look for \( u(t) \in V \), such that for each \( \phi \in V \)

\[
\frac{d}{dt} (u(t), \phi) + \mu (\nabla u(t), \nabla \phi) + b(u(t), u(t), \phi) = (f, \phi) \]

\[ u(0) = u_0. \] (2.29)

The trilinear form \( b \) is defined as

\[ b(u, v, w) = \int_{\Omega} u \cdot \nabla v \cdot w. \]

The solution of (2.29), which belongs to \( L^2(0, T; V) \cap L^\infty(0, T; [L^2(\Omega)]^d) \forall T > 0 \), is called weak solution of the Navier-Stokes problem. See [152] for a complete discussion.

A generalization of Navier-Stokes equations can be obtained by considering the Navier-Poisson constitutive equation for a Newtonian fluid [86]:

\[ \sigma(u, p) = (\lambda \text{tr}(D(u)) - p) I + 2 \mu D(u), \] (2.30)

In this case the momentum conservation equation reads:

\[
\rho \frac{\partial u}{\partial t} + \rho (u \cdot \nabla) u + \nabla p - \nabla \cdot (\mu \nabla u) \right) - (\lambda + \mu) \nabla (\nabla \cdot u) = \rho f \]

(2.31)

The fluid can now be compressible and this feature will be used in Section 4.2.1 to derive a mechanic model for fluid registration.
ALE formulation

When dealing with a moving continuum, different choices of reference system can be employed. If the coordinate system is based on the reference configuration \((\mathbf{x}, t)\), focusing the description on the material particle \(x\) and its evolution, the formulation is called Lagrangian, while if it is based on the current configuration \((x, t)\), setting the observation at a given point in the physical space, we obtain the Eulerian formulation. In solid mechanics the first approach is the preferred one, being the displacements relatively small, while in fluid mechanics the displacements are extremely large and the Eulerian framework is then preferred in this case.

In computational haemodynamics, a completely Eulerian description requires interpolation or projections steps for bringing the solution from one time framework to the other. This is sometimes possible (as in the Immersed Boundary Method) even if the overall accuracy of the method strongly relies on the accuracy of these operations. On the other hand, a complete Lagrangian perspective is not affordable for the large displacements of the fluid. A popular approach in this field is hence a third formulation which is in between the Lagrangian and the Eulerian ones, called Arbitrary Lagrangian Eulerian formulation (ALE). In this formulation the domain \(\Omega(t)\) is moving in time according to an auxiliary motion \(\hat{A} : \hat{\Omega} \times \mathbb{R}^+ \rightarrow \mathbb{R}^3\)

\[
\hat{A} : \hat{\Omega} \times \mathbb{R}^+ \rightarrow \mathbb{R}^3 \quad (\hat{x}, t) \rightarrow x = \hat{A}(\hat{x}, t),
\]

built from the desired evolution of \(\partial \Omega(t)\). The current configuration will be \(\Omega(t) = \hat{A}(\hat{\Omega}, t)\). Typically in blood flow problems the prescribed boundary motion follows the movement of the wall at fluid-solid interface while remains fixed at the artificial inlet and outlet sections.

The velocity of the computational domain is defined as

\[
\hat{w}(\hat{x}, t) = \frac{\partial \hat{A}}{\partial t}(\hat{x}, t), \quad \forall \hat{x} \in \hat{\Omega}.
\]

When describing the fluid motion in the ALE frame, it can be proven [53] that the conservation equations for an incompressible Newtonian fluid read

\[
\begin{align*}
\rho \frac{\partial u}{\partial t} + \rho ((u - \hat{w}) \cdot \nabla) u + \nabla p - 2\nabla \cdot (\mu D(u)) &= \rho f, \quad \text{in } \Omega(t), \ t > 0 \\
\nabla \cdot u &= 0, \quad \text{in } \Omega(t), \ t > 0,
\end{align*}
\]

where \(\Omega(t)\) is the time-dependent ALE domain and \(w\) is the domain velocity. The symbol \(\frac{\partial}{\partial t} \bigg|_A\) stands for the ALE time derivative, the rate of change of the considered quantity in a point that moves with the computational domain.

The convective term is modified according to the domain velocity. We may note that in case we turn to a Lagrangian description, the velocity domain is equal to the fluid domain and we recover the Lagrangian formulation, while if the domain is fixed, \(w = 0\) and we recover the Eulerian formulation.

The ALE formulation of the Navier-Stokes equations will be used in Sections 4.3 and 4.4 to simulate blood flow in compliant vessels.
2.2.2 Vessel wall mechanics

As mentioned before, to model solid mechanics it is more convenient to map back the conservation equations (2.23) and (2.25) to the reference domain. If we apply this procedure to (2.23) we get:

$$\frac{\partial \hat{J} \hat{\rho}}{\partial t} = 0, \quad \text{in } \hat{\Omega}, \ t > 0.$$  \hspace{1cm} (2.33)

The momentum conservation equation in Lagrangian formulation involves the first Piola-Kirchhoff tensor $\hat{\Pi}$. By mapping back to the reference configuration equation (2.25), we obtain

$$\hat{J} \hat{\rho} \frac{\partial^2 \hat{d}}{\partial t^2} - \nabla_{\hat{x}} \cdot \hat{\Pi} = \hat{J} \hat{\rho} \hat{f} \quad \text{in } \hat{\Omega}, \ t > 0$$  \hspace{1cm} (2.34)

which is referred to as equation of elastodynamics. Possible boundary conditions for this model are Dirichlet conditions, where the displacement is imposed to be equal to a given function, Neumann conditions in which the surface stress applied is prescribed, and Robin conditions, that can be used to minimize spurious wave reflections at the artificial boundaries of the domain (i.e. inlet and outlet sections of the vessel).

A simple model belonging to the hyperelastic materials class that can be used to model the vessels wall is the St. Venant-Kirchhoff model, which assumes the material to be homogeneous and isotropic. The deformation energy in this model is

$$\omega(\hat{E}) = \frac{\lambda}{2} (\text{tr} \hat{E})^2 + \mu \text{tr} \hat{E}^2,$$

where $\lambda$ and $\mu$ are the first and second Lamé coefficients. The stress-strain relation for a St. Venant-Kirchhoff material is consequently

$$\hat{\Sigma}(\hat{E}) = \lambda (\text{tr} \hat{E}) \hat{I} + 2\mu \hat{E}.$$  \hspace{1cm} (2.35)

The Lamé coefficients can be related to Young modulus $E$ and Poisson coefficient $\xi$, known from experimental measures, by

$$E = \mu \frac{3\lambda + 2\mu}{\lambda + \mu} \quad \text{and} \quad \xi = \frac{1}{2} \frac{\lambda}{\lambda + \mu}.$$

The St. Venant-Kirchhoff model provides a simple description of the wall mechanics. It does not describe in details the stress field inside the wall thickness, however it can be used to model the effects of the vessel on the haemodynamics in the fluid domain. The advantage of this choice is that numerical computations with this model are affordable.

We remark that more sophisticated models for the arterial wall are available in the literature, that take into account the microscopical structure of the tissue. We refer to [53,56,75] and to the literature cited therein for comprehensive reviews in this field.
3 Numerical methods for electrocardiology

In this Chapter we address the numerical solution of the electrophysiology problems introduced in Section 2. In particular in Section 3.1 we present the most popular numerical methods used in the literature to simulate the action potential propagation in the heart. We then focus on the description of the original numerical approaches that we propose to reduce the computational effort required by the simulations under consideration. In particular in Section 3.2 we discuss a model-based preconditioner developed ad-hoc for the solution of the Bidomain system, proving its promising spectral properties and showing some numerical results. In Section 3.3 we propose a hybrid model which is the composition of a Bidomain problem on a small part of the domain and a Monodomain problem in the remaining part. This partition of the domain can be chosen automatically during the simulation, by using a proper model adaptivity strategy based on an a posteriori model error estimator.

3.1 State of the art on numerical simulations for electrocardiology

In this Section we provide an overview on the most common numerical methods that are employed in the literature to solve the cardiac electrophysiology problem. The content of this introduction has been included in [28].

As we have shown in Section 2.1 a general electrocardiology problem is composed of a system of PDEs, describing the action potential propagation in the tissue, coupled with a system of ODEs, describing the cell membrane ionic behavior. In Section 3.1.1 we review some of the approaches used in the literature to achieve a time discretized problem.

After this step the PDEs need then to be discretized in space, to obtain an algebraic system. In Section 3.1.2 we address in particular the finite element method used to discretize in space the Bidomain and Monodomain models, since it is the approach employed in the present work.

The discretization steps produce a linear system to be solved at each time step. In Section 3.1.3 we report the most common approaches to solve this system, including preconditioning techniques and parallel implementations.

Finally in Section 3.1.4 we report the software packages available for simulation in this field.
3 Numerical methods for electrocardiology

3.1.1 Time discretization techniques

Time discretization can be carried out with explicit, implicit or semi-implicit (also called IM-EX or IMEX) approaches. In the former case, the solution at the new time step is straightforwardly computed as a function of the previous ones. In the implicit case, the new solution is still argument of a space dependent differential operator. Explicit methods have been used extensively [71,76,116,131,162], because they are easy to implement. However, even though the computational cost for each time step is low in an explicit method, the time step needs to be small to guarantee stability. Implicit schemes can be stable with longer time steps [17,76,99], but require solution of a non-linear system of equations at each time step, and so are more computationally expensive. A good compromise between these two methods is a semi-implicit IMEX scheme, in which some terms of the equations (e.g. the linear terms) are solved implicitly and the remaining terms (e.g. the non-linear terms) are solved explicitly. The stability constraints of IMEX methods are less restrictive than fully explicit methods but more than fully implicit methods. Since the linear terms are usually solved implicitly in an IMEX method, a solution of a linear system is required at each time step. The IMEX methods can be classified on the basis of the order of approximation they can achieve. First order methods are the most common [33,85], although higher order IMEX methods have been proposed. An extensive stability analysis of IMEX methods for solving the Bidomain equations coupled with the FitzHugh-Nagumo kinetic model can be found in [44], together with a comparison of computational costs.

The global system of PDEs and ODEs can be decoupled in several ways through the operator splitting technique, with the aim of improving the computational efficiency and reducing complex dependencies between the variables of the problem. In principle, these methods introduce a splitting error that should be quantified, unless the split steps are re-iterated until a suitable convergence condition is satisfied. In [129,132] an operator splitting technique on a finite difference formulation is used to solve the Bidomain problem coupled with a PDE representing the surrounding bath. In [125] a second order splitting technique has been proposed to separate the solution of the Monodomain PDE, describing the propagation in tissue, from the Luo-Rudy phase I ODEs, describing local membrane kinetics. The ODEs system was solved using the Rush-Larsen method [135], while the time advancing scheme for the PDE was an explicit forward Euler method. An alternative method for solving the ODEs system has been recently proposed in [118], which is a generalization of the popular Rush-Larsen method. An operator splitting technique has also been devised for solving the Bidomain equations coupled with a description of volume conduction in the torso [149]. In this study the authors proposed a general formulation of operator splitting, based on a \( \theta \)-rule, that can result in a second order scheme for a proper choice of the parameter. The ODEs system was solved using a third order Runge-Kutta scheme. The ODEs system itself can be decomposed using operator splitting methods. In [122] a numerical method is proposed, splitting each one of the two Bidomain PDEs and the ODEs system method. Both the PDEs are discretised via a forward Euler scheme while the ODEs system is solved implicitly. In [163] the authors use for decoupling the Bidomain system a three-step operator splitting that has
been proven to be unconditionally stable. Operator splitting techniques separating the non-linear part of the ODEs system from the linear one allow the use of different numerical schemes to address sub-systems of equations with different complexity and time scales.

**Time discretization schemes in the present work**

We now describe the time discretization schemes employed throughout this Chapter, for solving the Bidomain and Monodomain models coupled with Rogers-McCulloch ionic model or Luo-Rudy I ionic model.

We refer hereafter to the formulation of the Bidomain (2.12) and of the Monodomain (2.19) models. Nevertheless the same time discretization scheme can be applied to the non-symmetric formulation of the Bidomain that we will introduce in Section 3.2 and for the Hybridomain model proposed in Section 3.3.

In particular, denoting with $\delta t$ the time step, with superscript $n$ the variables at time $t = n\delta t$ and with superscript $n+1$ the variables at time $t = (n+1)\delta t$, the time discrete Bidomain problem reads

$$
\begin{align*}
\chi_C m^{n+1} - u^n - \nabla \cdot (D_i \nabla u_i^{n+1}) + \chi I_{ion}(u^*, w^*) &= I_{app}^{i,n+1} \\
- \chi_C m^{n+1} - u^n - \nabla \cdot (D_e \nabla u_e^{n+1}) - \chi I_{ion}(u^*, w^*) &= -I_{app}^{e,n+1}
\end{align*}
$$

(3.1)

The discretization of term $I_{ion}(u^*, w^*)$ depends on the expression of the ionic current, hence on the considered ionic model and will be addressed later on in this Section.

The same approximation is employed for the Monodomain equation (2.19), which reads:

$$
\begin{align*}
\chi_C m^{n+1} - u^n - \nabla \cdot (D_M \nabla u^{n+1}) + \chi I_{ion}(u^*, w^*) &= I_{app}^{n+1} \\
u^0 &= u_0 \\
n^T D_M \nabla u^{n+1} &= 0
\end{align*}
$$

(3.2)

Let us now detail the approximation of the term $I_{ion}(u^*, w^*)$, considering the ionic models employed in this work.

When a tissue model is coupled with the Rogers-McCulloch ionic model, the ordinary differential equation in (2.3) is solved by discretizing explicitly the transmembrane potential while considering as implicit every other term in the equation:

$$
\frac{w^{n+1} - w^n}{\delta t} = \eta_2 \left( \frac{u^n}{v_p} - \eta_3 w^{n+1} \right)
$$

43
3 Numerical methods for electrocardiology

The ionic current $I_{ion}(u^*, w^*)$, which in this case depends on a scalar recovery variable $w$, can be discretized by considering the transmembrane potential $u^*$ at the previous time step and the recovery variable at the current one, obtaining:

$$I_{ion}(u^n, w^{n+1}) = G u^n \left(1 - \frac{u^n}{v_{th}}\right) \left(1 - \frac{u^n}{v_p}\right) + \eta_1 u^n w^{n+1}$$

The scheme obtained by applying these approximations has been employed for instance in [33] and is a semi-implicit first order scheme.

In the numerical tests in which we simulate the coupling of a tissue model with the Luo-Rudy I ionic model, we employ the first-order semi-implicit Rush-Larsen scheme described in [135]. We present the main idea of this approach considering a single non-linear ODE from system (2.6), which can be expressed in the general form

$$\begin{align*}
\frac{dw_k}{dt} &= a_k(t, w) w_k + b_k(t, w), \quad k = 1, ..., 6, \quad t \in (0, T] \\
w_k(0) &= w_{k,0},
\end{align*}$$

(3.3)

where $a_k$ and $b_k$ represent the coefficients of the equation, depending on the unknown vector $w$. The Rush-Larsen method consists in considering functions $a_k(t, w)$ and $b_k(t, w)$ constant on the interval $(t^n, t^{n+1}]$ and equal to $a^n_k$ and $b^n_k$. With this approximation the ordinary differential equations become linear and can be solved exactly. The solution obtained after this process is proven to be a first order approximation of the solution of the original non-linear system (3.3).

As reported in [117], this explicit scheme for solving the ODEs system allows us to take a time-step significantly greater than in the Forward Euler (FE) scheme, still avoiding numerical instability. Moreover the accuracy can be improved to a second order, by extending the Rush-Larsen scheme, as shown in [118].

3.1.2 Space discretization

The most popular space discretization methods in electrophysiology modeling are the finite difference method and the finite element method. However in some cases also the finite volume method [155], the boundary element method [50], mesh free methods [25] and spectral methods [20, 113] have been used.

In this work the time discretized systems (3.1) and (3.2), as well as their modifications that will be presented in the next Sections, have been discretized in space using finite elements approximations. The domain of the problem is discretized in a computational grid $\mathcal{T}_h$ composed of tetrahedral elements and the ODEs systems in the ionic models are solved pointwise on the grid nodes. The unknowns $(u_i, u_e)$ are set in the finite element space $\mathcal{V}_h$, which is an approximation of the Sobolev space $\mathcal{V}$, introduced to define equation (2.14). In this work each component of $\mathcal{V}_h$ is the space of piecewise linear continuous functions on the grid $\mathcal{T}_h$.

The space-time discretized formulation of the considered PDEs model, obtained in this way, results in a linear system of equations that can be solved using standard numerical
3 Numerical methods for electrocardiology

linear algebra methods or ad hoc methods, in a time advancing scheme. For the detailed composition of such linear systems we refer to Section 3.2, where the matrices involved in the considered formulations are described.

3.1.3 Solution of the linear system

Most numerical approaches to solve tissue electrophysiology models result in a linear system of equations.

The most widely used approach to solve these systems is to employ iterative solvers. To speed up their convergence, several types of preconditioners can be used. One common option is to compute an incomplete LU (or an incomplete Cholesky) factorisation. This approach has been shown to be effective on serial architectures [44]. A comparison using a parallel architecture has shown that a CG solver preconditioned by a domain decomposition parallelisation of an incomplete LU factorisation to be twice as fast as the same solver preconditioned with a Jacobi preconditioner [123]. In another study [31] a CG solver preconditioned by Symmetric Successive Over-Relaxation preconditioner was used for solving the eikonal model, which is a reduced model able to track the propagation of the action potential wavefront (see [32]). We will describe in Section 3.2 a specific preconditioner for the Bidomain problem, consisting of a model-based block triangular preconditioner obtained after a proper reformulation of the Monodomain model [60]. This approach reduces the CPU time required by the incomplete LU preconditioned CG solver on serial architectures by about 50%.

Multigrid methods have been employed for the solution of Mono- and Bidomain models [82]. They are based on the combination of an iterative solver running on fine computational grids of the domain and a direct solver running on a coarse grid of the same domain [68]. By repeating these steps both the high and the low frequencies of the residual of the linear system can be reduced within a specified tolerance. Multigrid methods have also been applied as preconditioner for other solvers such as CG, and may be more effective than incomplete LU preconditioned CG in parallel implementations [122, 148, 164, 165]. This preconditioner can be used both with serial and parallel implementation and the best number of levels of the multigrid strategy depends on the number of processors and on the available memory [163].

In [115] the authors propose and analyze a Multilevel Additive Schwarz preconditioner for the Bidomain system to overcome the multiplicative nature of multigrid methods, preventing them to be fully parallelisable. This preconditioner has been shown to be both scalable and to offer a potentially optimal approach on structured meshes.

Various parallel implementations of solvers have been developed and applied for simulations of tissue electrophysiology. For example, some simulators [123] are based on OpenMP[1] an implementation of multithreading that supports shared memory architectures. The main advantage is that it can be easily developed, since the serial code needs only to be slightly modified or can use indirectly OpenMP by interfacing with specific linear algebra packages able to deal with OpenMP. The main drawback is that

---

the parallel scalability is limited and dependent of computer architecture. An alternative approach is based on MPI\textsuperscript{2}, a message-passing application programming interface that allows many computers to communicate with one another. In this case, specific code has to be developed to deal with MPI, but high scalability can be achieved. MPI based code can run on both distributed and shared memory architectures. Examples of this programming strategy applied to tissue electrophysiology simulations can be found in [33, 115]. An intermediate approach involves developing hybrid code that can take advantage of developments in the architecture of high performance computers [15].

### 3.1.4 Available software packages

In the last years several software packages have been developed to numerically simulate the action potential propagation in the myocardial tissue. In this subsection we list the most popular ones, without claiming completeness, reporting the main features for each of them. We refer to the related bibliography for more detailed information.

A collaboration between the University of Calgary and Medical University of Graz has developed the Cardiac Arrhythmia Research Package CARP\textsuperscript{3}, a parallel simulator that implements an operator splitting method applied to the finite elements approximation of the Bidomain problem. The solution of the ionic model is carried out by a specific library called LIMPET. CARP also includes a mesh generator and a visualization tool. The Oxford University Computing Lab has developed the Cancer Heart And Soft Tissue Environment CHASTE\textsuperscript{4}, a multi-purpose software that includes a Monodomain and Bidomain simulator. It employs PyCml to interface with CellML repository for ionic models, and is able to generate mathematical models for the fiber orientation. It is able to generate simple geometry computational grids and to read tetrahedral meshes. It supports the use of parallel architectures and it is available for download. The Continuity\textsuperscript{5} software, developed by McCulloch group at University of California San Diego and available for download, includes an electrophysiology module able to solve the Monodomain problem coupled with popular ionic models, including Beeler-Reuter, Luo-Rudy and FitzHugh-Nagumo ones. It also provides an interface for implementing different cellular models and an interface for using parallel solvers. CardioWave\textsuperscript{6} is a modular electrophysiology simulator built up at Duke University to solve the Bidomain equations in a finite elements framework. The MembraneModule part takes into account the ionic model while TimeIntegrationModule and LinearSolverModule are in charge for the numerical solution. The software is available for download. Another software package, named acCELLerate and developed by researchers at Universität Karlsruhe, is able to simulate the Bidomain model employing a finite elements discretization. This software makes use of an operator splitting algorithm to achieve modularity and is strongly based on PETSc\textsuperscript{7}.

\textsuperscript{2}Message Passing Interface, http://www.mpi-forum.org/
\textsuperscript{3}http://carp.meduni-graz.at/
\textsuperscript{4}http://web.comlab.ox.ac.uk/chaste/
\textsuperscript{5}http://www.continuity.ucsd.edu/
\textsuperscript{6}http://cardiowave.duke.edu/
\textsuperscript{7}http://www.mcs.anl.gov/petsc/petsc-2/
library for parallel implementation. acCELLerate uses Krylov subspace methods and preconditioner provided by PETSc to solve the linear system obtained after the discretization. More details can be found in [83]. The software package KARDOS\(^8\), developed at Konrad-Zuse-Zentrum für Informationstechnik Berlin (ZIB), is a software package for solving nonlinear parabolic systems. The one-dimensional version of the code can be downloaded from the website. KARDOS has been used to implement a time and space adaptivity in a finite elements setting, applied to the Monodomain and Bidomain problem coupled with FitzHugh-Nagumo or Luo-Rudy phase I ionic models. Finally, LifeV\(^9\) is a finite element library used for medical and industrial applications and developed at Ecole Polytechnique Fédérale de Lausanne, Institute National de Recherche en Informatique et en Automatique, Politecnico di Milano and Emory University that have been used for performing the numerical simulations described in this thesis. Concerning electrocardiology, it features numerical solvers for the Monodomain and the Bidomain problems, and it is modular with respect to the cellular membrane ODEs systems used. So far Rogers-McCulloch and Luo-Rudy phase I models have been implemented. Both the serial version and the parallel version of the code, are based on Trilinos software package\(^{10}\) for the solution of linear systems. More details about LifeV library can be found in Section 5.

3.2 Model preconditioning

In Section 2.1 we already mentioned that, though the Bidomain model is the most accurate available in the literature to simulate the action potential propagation in the myocardium, the degenerate parabolic nature of its discretized system implies high computational costs to compute the numerical solution. Replacing it with the Monodomain model, is not completely satisfactory, since the latter is derived under a quite unrealistic assumption and some relevant patterns in the potential propagation can be missed by this simplified model.

Since the Monodomain problem is simpler to solve than the Bidomain but its discretized version has nevertheless a similar structure, we propose to use an Extended Monodomain model as a preconditioner in solving a suitable non-standard non-symmetric reformulation of the Bidomain system.

We present hereafter the formulation of the Monodomain preconditioner, we review its spectral properties providing bounds for the eigenvalues and for the condition number of the preconditioned matrix that demonstrate its optimality and we show some numerical results on both simplified and real geometries representing the left ventricle. The most part of this work has been recently published in [60].

---

\(^8\)http://www.zib.de/Numerik/numsoft/kardos/
\(^9\)http://www.lifev.org/
\(^{10}\)http://trilinos.sandia.gov
3 Numerical methods for electrocardiology

3.2.1 Monodomain preconditioner for the Bidomain system

To use the Monodomain model as a preconditioner for the Bidomain system we need to reformulate both the systems in a suitable way.

The standard Bidomain problem in symmetric form (2.12) can be manipulated by applying a linear combination to its equations, with coefficients \( \lambda \) and \(-\lambda\), the same used to obtain the Monodomain equation (2.19). We obtain then a new equation in terms of the transmembrane and the extracellular potentials \( u \) and \( u_e \) that constitutes the first equation of the new formulation of the Bidomain. The second equation is obtained by summing up the two equations in (2.12).

Setting \( I_{\text{app}} = \lambda I_{\text{app}}^i + I_{\text{app}}^e \) and \( \tilde{I}_{\text{app}} = I_{\text{app}}^i - I_{\text{app}}^e \), the resulting system reads:

\[
\begin{align*}
\chi C_m \frac{\partial u}{\partial t} - \nabla \cdot \left( \frac{\lambda D_i}{1 + \lambda} \nabla u \right) - \nabla \cdot \left( \frac{\lambda D_i - D_e}{1 + \lambda} \nabla u_e \right) + \chi I_{\text{ion}}(u, w) &= I_{\text{app}} \\
- \nabla \cdot [D_i \nabla u + (D_i + D_e) \nabla u_e] &= \tilde{I}_{\text{app}}.
\end{align*}
\]

(3.4)

Hereafter we will refer to (3.4) as “non-symmetric formulation” of Bidomain model. We point out that formulation (3.4) is non-standard and has been introduced in view of our preconditioning technique.

In order to match the size of the Bidomain problem, also the Monodomain problem needs a proper reformulation which extends the original position. It consists in the same linear combination leading to (3.4), combined with the assumption \( D_e = \lambda D_i \):

\[
\begin{align*}
\chi C_m \frac{\partial u}{\partial t} - \nabla \cdot \left( \frac{\lambda D_i}{1 + \lambda} \nabla u \right) + \chi I_{\text{ion}}(u, w) &= I_{\text{app}} \\
- \nabla \cdot [D_i \nabla u + (1 + \lambda)D_i \nabla u_e] &= \tilde{I}_{\text{app}}.
\end{align*}
\]

(3.5)

System (3.5) is lower triangular, where the first equation (the “genuine” Monodomain model) is independent of \( u_e \). In view of its use as a preconditioner, however, there is no reason for retaining the simplifying Monodomain assumption \( \lambda D_i = D_e \) in the second equation so we finally resort to

\[
\begin{align*}
\chi C_m \frac{\partial u}{\partial t} - \nabla \cdot \left( \frac{\lambda D_i}{1 + \lambda} \nabla u \right) + \chi I_{\text{ion}}(u, w) &= I_{\text{app}} \\
- \nabla \cdot [D_i \nabla u + (D_i + D_e) \nabla u_e] &= \tilde{I}_{\text{app}}.
\end{align*}
\]

(3.6)

Observe that our formulation of the Monodomain model comes immediately from the non-symmetric Bidomain (3.4) when the differential term in \( u_e \) in the first equation is dropped. As for the Bidomain model, also in the Extended Monodomain model (3.5) (or (3.6)) the extra cellular potential \( u_e \) is defined only up to a function of time. We will fix such function by requiring that \( u_e \) has zero mean.

3.2.2 Numerical discretization

Let \( \delta t \) be the (constant) time step of the discretization. We denote with superscript \( n \) the variables computed at time \( t^n = n \delta t \). Moving from time step \( t^n \) to \( t^{n+1} \) the semi-implicit
time-discretization of Bidomain equations (3.4) reads

\[
\begin{cases}
\chi C_m \frac{u^{n+1} - u^n}{\delta t} - \nabla \cdot \left( \frac{\lambda D_i}{1 + \lambda} \nabla u^{n+1} + \frac{\lambda D_i - D_e}{1 + \lambda} \nabla u_e^{n+1} \right) = I^n & \text{in } \Omega \\
- \nabla \cdot \left[ D_i \nabla u^{n+1} + (D_i + D_e) \nabla u_e^{n+1} \right] = \tilde{I}^{\text{app}} & \text{in } \Omega \\
u^0(x) = u_0(x) & \text{in } \Omega \\
n^T D_i (\nabla u^{n+1} + \nabla u_e^{n+1}) = 0 & \text{on } \partial \Omega \\
n^T D_e \nabla u_e^{n+1} = 0 & \text{on } \partial \Omega 
\end{cases}
\]  

(3.7)

where we have set \( I^n = \tilde{I}^{\text{app}} - \chi I_{\text{ion}}(u^n, w^{n+1}) \), the latter term including the selected model for ionic current. Concerning the spatial approximation, as anticipated in Section 3.1.2, we discretize the domain \( \Omega \) with a triangulation \( T_h \) and we build a finite element space \( V_h \) approximating \( H^1(\Omega) \) on \( T_h \), in which we will look for the approximate solution \( u_h \) and \( u_e \). In this work \( V_h \) is the space of piecewise linear continuous functions on \( T_h \), and we denote by \( \Phi = \{ \phi_j \}_{j=1}^{N_h} \) a basis for \( V_h \). Well-posedness of the discrete problem and convergence analysis for the Rogers-McCulloch model are carried out in [141].

Let us denote by \( M \) the mass matrix with entries \( M_{ij} = \sum_{K \in T_h} \left( \frac{1}{1 + \lambda} \right) |K| \), and by \( K \) the stiffness matrices with \( K_{ij} = \sum_{K \in T_h} \left( \frac{\lambda D_i}{1 + \lambda} \right) |K|, \phi_i, \phi_j \in \Phi \). The unknowns of the fully discrete problem are represented by vectors \( u \) and \( u_e \), storing the nodal values of \( u_h \) and \( u_e \), respectively, we let \( f \) and \( g \) denote the discretization of the forcing terms, and we set

\[
B_{uu} = \frac{M}{\delta t} + \frac{\lambda K_i}{1 + \lambda} \quad B_{ue} = \frac{\lambda K_i}{1 + \lambda} - \frac{K_e}{1 + \lambda} \quad B_{eu} = K_i \quad B_{ee} = K_i + K_e.
\]

At step \( n+1 \) we solve

\[
B_{\text{NS}} u_{\text{NS}}^{n+1} = f_{\text{NS}},
\]

(3.8)

where

\[
B_{\text{NS}} = \begin{bmatrix}
B_{uu} & B_{ue} \\
B_{eu} & B_{ee}
\end{bmatrix}, \quad u_{\text{NS}} = \begin{bmatrix} u \\ u_e \end{bmatrix}, \quad f_{\text{NS}} = \begin{bmatrix} f \\ g \end{bmatrix}.
\]

As a preconditioner for (3.8) we select

\[
M_{\text{NS}} = \begin{bmatrix}
B_{uu} & 0 \\
B_{eu} & B_{ee}
\end{bmatrix}
\]

(3.9)

which corresponds to the discrete operator associated to the Extended Monodomain problem (3.6), suitably discretized in time. This form highlights that our model-based preconditioner results in a block Gauss-Seidel preconditioner for the non-symmetric Bidomain system, featuring \( u \) and \( u_e \). However, we will still refer to the Monodomain-based interpretation, which allows us to gain insight on its effectiveness. In principle, the same approach based on the block-triangular preconditioning could be applied also to the symmetric formulation of the Bidomain system in terms of \( u_i \) and \( u_e \). However, this choice proves to be ineffective, as we show at the end of Section 3.2.4.
Linear solver  Since the matrix in (3.8) is not symmetric, we adopt a Krylov iterative solver. More precisely, in order to reduce the CPU time, we refer to a flexible strategy, corresponding to solve inaccurately the preconditioner by an iterative method with a coarse tolerance. In this case, the actual preconditioner depends on the current iteration. A Flexible GMRES (FGMRES) with a right preconditioner (see [136]) needs then to be used accordingly. By extension, we use the same right preconditioning approach also for non flexible GMRES with an accurate solution of the preconditioner.

The implementation of the Monodomain preconditioner requires to solve system

$$M_{NS} z = v$$

, where $z = [z^1, z^2]^T$ and $v = [v^1, v^2]^T$. To this aim, we exploit the lower triangular structure of $M_{NS}$, solving the sequence of the systems

$$B_{uu} z^1 = v^1 \quad b = v^2 - B_{ee} z^1 \quad B_{ee} z^2 = b.$$ (3.10)

Since systems in $B_{uu}$ and $B_{ee}$ are symmetric, we solve them using ILU preconditioned conjugate gradient method. Notice that both matrices $B_{NS}$ and $M_{NS}$ defined in (3.8) and (3.9) respectively are singular, their kernel being given by $\text{span}\{[0, 1]^T\}$. In particular, we solve the singular systems by an iterative method, as this is a reliable strategy for elliptic problem with homogeneous Neumann boundary conditions [13]. After the solution of non-symmetric Bidomain system (3.8), we force the average of $u_e$ to be zero at each time iteration.

3.2.3 Spectral properties of the preconditioner

By performing an analysis similar to the one carried out in [59] and [5] for advection diffusion and Maxwell problems respectively, we show that the eigenvalues of the preconditioned matrix are bounded by quantities not depending on the space discretization parameter $h$.

We report hereafter the general setting of the Fourier analysis we employed and the results obtained, suggesting [60] for the details on the computation. We rather report here the computation of the singular values of the preconditioned matrix and the resulting condition number of the preconditioned operator, since this part of the work is not reported in the cited paper.

To analyze the preconditioner, without loss of generality we assume the reference frame to have the first component aligned with the longitudinal axis of the fibers, so that, the diffusion tensors are diagonal, under axial isotropy assumption, and we set the problem in an unbounded domain $\Omega \equiv \mathbb{R}^3$.

Let us first introduce the basis of this analysis: denoting by $k_1$, $k_2$ and $k_3$ the dual frequency variables, the Fourier transform of $w(x, y, z) = u(x, y, z), u_e(x, y, z)$ reads

$$\mathcal{F} : w(x, y, z) \mapsto \hat{w}(k_1, k_2, k_3) = \int \int \int_{\mathbb{R}^3} e^{-i(k_1 x + k_2 y + k_3 z)} w(x, y, z) \, dx \, dy \, dz.$$
We can also introduce the continuous operators

\[ B : [H^1(\Omega)]^2 \to [H^{-1}(\Omega)]^2 \]

and

\[ M : [H^1(\Omega)]^2 \to [H^{-1}(\Omega)]^2 , \]

that are the continuous counterparts associated with the discretized problems (3.7) and (3.6).

Action of operators \( B \) and \( M \) can now be expressed for any \( u \in [H^1(\Omega)]^2 \) by means of the inverse Fourier transform, namely

\[ B \hat{u} = \mathcal{F}^{-1}(B \hat{u}) \]

\[ M \hat{u} = \mathcal{F}^{-1}(M \hat{u}) \]

where \( B \) and \( M \) represent in the frequency domain the operators \( B \) and \( M \), respectively.

In this case, asymptotic requirements for the Fourier transformability (see [133]) of the extracellular potential automatically fix the arbitrary function of time, so no arbitrariness is anymore present. For the sake of notation, time index is dropped hereafter.

By applying the Fourier transformation to the Bidomain problem (3.7) and to the Extended Monodomain one we obtain the matrix associated with the preconditioned operator in the frequency domain:

\[
P(\xi, \eta) \equiv [M(\xi, \eta)]^{-1} B(\xi, \eta) = \begin{bmatrix}
1 & \alpha(\xi, \eta) \\
0 & 1 - \frac{\xi}{\xi + \eta} \alpha(\xi, \eta)
\end{bmatrix}.
\]

(3.11)

where we have defined \( \xi = \sigma^2_l k^2_1 + \sigma^2_t k^2_2, \eta = \sigma^2_l k^2_1 + \sigma^2_t k^2_3 \), with \( k^2 = k^2_2 + k^2_3 \), and

\[
\alpha(\xi, \eta) = \frac{\delta t}{1 + \lambda} \frac{[\lambda \xi - \eta]}{1 + \frac{1}{1 + \lambda} \delta t \xi}.
\]

(3.12)

The effectiveness of the preconditioned operator has been analyzed in a domain in the \((\xi, \eta)\) plane depending on the minimum and maximum values for \( \lambda, \lambda_m \) and \( \lambda_M \) defined in (2.17) and on the maximal frequencies supported by the numerical grid (of the order of \( \pi/h \)). As the maximal supported frequencies tend to infinity this domain covers the angular sector \( \mathcal{S} = \{ \lambda_m \xi \leq \eta \leq \lambda_M \xi \} \setminus \{(0,0)\} \).

The eigenvalues of \( M^{-1} B \) are given by

\[ \gamma_1(\xi, \eta) = 1 \quad \gamma_2(\xi, \eta) = 1 - \frac{\xi}{\xi + \eta} \alpha(\xi, \eta) = \frac{1 + \delta t \xi}{1 + \frac{\xi}{\eta + 1}} \frac{\frac{\xi}{\eta} + 1}{1 + \delta t \xi} \]

(3.13)

It is proved in [60] that in the considered domain, the following bounds on \( \gamma_2 \) hold:

\[
\frac{1}{\lambda_m + 1} \leq \min_{(\xi, \eta) \in T} \gamma_2(\xi, \eta) \leq \gamma_2(\xi, \eta) \leq \max_{(\xi, \eta) \in T} \gamma_2(\xi, \eta) \leq \frac{1}{\lambda_M + 1}.
\]

(3.14)
Therefore the eigenvalues of the preconditioned matrix are bounded by constants depending only on the anisotropy ratio in the coefficients of the Bidomain problem and on the chosen \( \lambda \) and are independent of the Fourier variables representing the discretization parameter of the grid.

The previous analysis suggests some further considerations on \( \lambda \). Beyond physical meaning, \( \lambda \) can be considered here as a parameter to be selected for enhancing the convergence of the preconditioned iterations. We plot in Figure 3.1 the distribution of the generalized eigenvalues \( \omega \) of the matrices \( B_{NS} \) and \( M_{NS} \) (\( B_{NS}v = \omega M_{NS}v \)), computed with Matlab\(^{11}\) for two different mesh sizes. Taking the values for \( \sigma_1 \tau \) and \( \sigma_t \tau \) proposed in [33], we have \( \lambda_m = 0.6667 \) and \( \lambda_M = 4.2868 \). We consider the three cases, namely \( \lambda = \lambda_m \), \( \lambda = \lambda_M \), and \( \lambda = 1.3 \), which is the value used in the numerical simulations of Section 3.2.4. Corresponding bounds for \( \gamma_2 \) are:

\[
1 \leq \gamma_2 \leq 2.02706 \quad \text{when} \quad \lambda = \lambda_m, \quad 0.49332 \leq \gamma_2 \leq 1 \quad \text{when} \quad \lambda = \lambda_M \quad \text{and} \quad 0.70771 \leq \gamma_2 \leq 1.43458 \quad \text{when} \quad \lambda = 1.3.
\]

Notice that the spectrum spreads out as the mesh parameter \( h \) decreases, however predicted bounds for the eigenvalues are fulfilled in both the cases. The choice of \( \lambda = 1.3 \), that we empirically tuned for minimizing computational costs, leads to a good clustering of the spectrum around 1.

We also estimate the conditioning of the preconditioned problem, by analyzing the singular values of the non-symmetric preconditioned matrix \( P(\xi, \eta) \).

We therefore analyze the eigenvalues of the matrix \( [P(\xi, \eta)]^T P(\xi, \eta) \), which is given by

\[
[P(\xi, \eta)]^T P(\xi, \eta) = \begin{bmatrix}
1 & \alpha(\xi, \eta) \\
\alpha(\xi, \eta) & [\gamma_2(\xi, \eta)]^2 + [\alpha(\xi, \eta)]^2
\end{bmatrix}.
\]

From (3.13), \( \alpha \) can be expressed as

\[
\alpha(\xi, \eta) = \left(1 + \frac{\eta}{\xi}\right) \left[1 - \gamma_2(\xi, \eta)\right],
\]

and the eigenvalues of \( [P(\xi, \eta)]^T P(\xi, \eta) \) are given by

\[
\mu^\pm(\xi, \eta) = \frac{1}{2} \left[1 + [\gamma_2(\xi, \eta)]^2 + \left(1 + \frac{\eta}{\xi}\right)^2 \left(1 - \gamma_2(\xi, \eta)\right)^2\right] ^{\pm \frac{1}{2} \sqrt{1 + [\gamma_2(\xi, \eta)]^2 + \left(1 + \frac{\eta}{\xi}\right)^2 \left(1 - \gamma_2(\xi, \eta)\right)^2}} - 4[\gamma_2(\xi, \eta)]^2
\]

Both eigenvalues are positive and we have \( \mu^- < \mu^+ \), then

\[
\max_{(\xi, \eta)} \left[\mu^-(\xi, \eta), \mu^+(\xi, \eta)\right] = \max_{(\xi, \eta)} \mu^+(\xi, \eta)
\]

\[
\min_{(\xi, \eta)} \left[\mu^-(\xi, \eta), \mu^+(\xi, \eta)\right] = \min_{(\xi, \eta)} \mu^-(\xi, \eta)
\]

\(^{11}\)http://www.mathworks.com/
3 Numerical methods for electrocardiology

Figure 3.1: Spectra of the preconditioned problem for different mesh sizes: 5272 nodes (top) and 12586 nodes (bottom). The dashed-dotted lines highlight clustering of the eigenvalues around 1 for different values of $\lambda$.

Thanks to the bounds on $\gamma_2$ reported in (3.14), and to the fact that $\lambda_m \leq \frac{\eta}{\xi} \leq \lambda_M$, we get the bound for the maximum eigenvalue:

$$
\mu^+(\xi, \eta) < 1 + [\gamma_2(\xi, \eta)]^2 + \left(1 + \frac{\eta}{\xi}\right)^2 \left[1 - \gamma_2(\xi, \eta)\right]^2
$$

$$
\leq 1 + \left[\frac{1}{\lambda_M} + 1\right]^2 + (1 + \lambda_M)^2 \left[1 - \gamma_2(\xi, \eta)\right]^2.
$$

(3.16)
We now turn our attention on the term \([1 - \gamma_2(\xi, \eta)]^2\): we have, from (3.12) and (3.13),
\[
\gamma_2(\xi, \eta) \leq 1 \iff \eta \leq \lambda \xi
\]
In particular, we have for \(\eta \leq \lambda \xi\)
\[
1 - \gamma_2(\xi, \eta) \leq \left[ \frac{1}{\lambda + 1} - \frac{1}{\lambda_m + 1} \right] \frac{\Delta t \xi}{1 + \Delta t \xi} \frac{1}{\lambda + 1}
\]
and for \(\eta > \lambda \xi\)
\[
\gamma_2(\xi, \eta) - 1 \leq \left[ \frac{1}{\lambda_M + 1} - \frac{1}{\lambda + 1} \right] \frac{\Delta t \xi}{1 + \Delta t \xi} \frac{1}{\lambda + 1}
\]
The term in \(\xi\) is monotone increasing thus it can be bounded from above by its limit as \(\xi \to \infty\):
\[
\frac{\Delta t \xi}{1 + \Delta t \xi} \frac{1}{\lambda + 1} \leq \lim_{\xi \to \infty} \frac{\Delta t \xi}{1 + \Delta t \xi} \frac{1}{\lambda + 1} = \frac{1}{\lambda + 1}.
\]
In conclusion we have
\[
|1 - \gamma_2(\xi, \eta)| \leq m_l \equiv \max \left\{ \left[ \frac{1}{\lambda_M + 1} - \frac{1}{\lambda + 1} \right] , \left[ \frac{1}{\lambda_m + 1} - \frac{1}{\lambda + 1} \right] \right\} = \begin{cases} 
\frac{1}{\lambda - \lambda_m} & \lambda_m \leq \lambda \leq \lambda_* \\
\frac{1}{\lambda_M + 1} & \lambda_* \leq \lambda \leq \lambda_M,
\end{cases}
\]
where
\[
\lambda_* = \frac{\lambda_m + \lambda_M + 2 \lambda_m \lambda_M}{2 + \lambda_m + \lambda_M}.
\] (3.17)
The expression of \(m_l\) is depicted in Figure 3.2

Therefore, we get
\[
\mu^+(\xi, \eta) \leq \mu^*_+ = \begin{cases} 
1 + \left[ \frac{1}{\lambda + 1} \right]^2 + \left[1 - \frac{\lambda_M}{\lambda} \right]^2 & \lambda_m \leq \lambda \leq \lambda_* \\
1 + \left[ \frac{1}{\lambda_M + 1} \right]^2 + \left[ \frac{1 + \lambda_M}{1 + \lambda_m} \right]^2 \left[1 - \frac{\lambda_m}{\lambda} \right]^2 & \lambda_* \leq \lambda \leq \lambda_M.
\end{cases}
\] (3.18)
To estimate the minimum of \(\mu^-(\xi, \eta)\) we note that
\[
\mu^-(\xi, \eta) = \frac{[\gamma_2(\xi, \eta)]^2}{\mu^+(\xi, \eta)},
\]
thus the minimum eigenvalue is bounded by

$$\mu^-(\xi, \eta) \geq \frac{\gamma_2^2}{\mu^+}$$  \hspace{1cm} (3.19)

where, thanks to (3.14), we set

$$\gamma_2 = \frac{1}{\lambda} + \frac{1}{\lambda + 1}.$$  \hspace{1cm} (3.20)

The condition number $K(P)$ in norm 2 of the continuous preconditioned problem can be expressed as function of the singular values $\sigma_{\pm}$ of $P(\xi, \eta)$, which are defined as the square root of $\mu_{\pm}$:

$$K(P) = \sigma_+(P)/\sigma_-(P).$$

Thus, after our analysis on $\mu_{\pm}$ we get that $K(P)$ is bounded by a value depending only on the coefficients of the problem and the parameter $\lambda$:

$$K(M^{-1}B) \leq \frac{\mu^+_*}{\gamma_2^*}.$$  \hspace{1cm} (3.21)

We conclude that the conditioning of the preconditioned problem at the discrete level is independent of the mesh size $h$ and the Monodomain preconditioner is consequently optimal with respect to the mesh size. Optimality of the preconditioner is particularly significant for 3D simulations on fine discretization of real geometries.

We report in the Figure 3.3 the upper bound on the condition number as a function of $\lambda$.

Optimality of the preconditioner is expected to be the trade-off between two circumstances. On the one hand, our non-symmetric formulation of the Bidomain problem is
3 Numerical methods for electrocardiology

Figure 3.3: Upper bound of the conditioning of the preconditioned problem as a function of parameter $\lambda$. The vertical dashed line highlights the value of $\lambda^\ast$.

such that the block of the original Bidomain matrix dropped in the preconditioner can be made quantitatively small by an appropriate selection of parameter $\lambda$. On the other hand, in parabolic problems the most unfavourable part in terms of dependence of the condition number on the mesh size is the elliptic one (stiffness matrix) (see [54]). The proposed preconditioner actually retains the elliptic core of the Bidomain system.

3.2.4 Numerical results

Numerical results presented hereafter refer to the 3D Bidomain problem on different geometries: a slab, a truncated ellipsoid, representing a simplified ventricular geometry, and a real geometry reconstructed from SPECT images as described in Section 4.1.2 (see Figure 3.4). All the geometries are completed with an analytical representation of the fiber orientation as detailed in [33]. Simulations on the slab and on the real ventricle use the Luo-Rudy Phase I model, while in the case of the truncated ellipsoid the Bidomain model is coupled with either the Rogers-McCulloch or the Luo-Rudy Phase I ionic models. We consider the parameters listed in [91] for the Luo-Rudy ionic model and the parameters in [33] to set the Rogers-McCulloch model and the Bidomain one. We report the used parameters in Appendix A.1 and A.2. Throughout this section, unless differently stated, we use a time step $\delta t = 0.5$ ms for the Rogers-McCulloch ionic model, while $\delta t = 0.1$ ms is needed for the Luo-Rudy one, in order to solve the ionic problem accurately enough.

Numerical simulations are carried out with ad hoc solvers implemented LifeV software [1], using Trilinos\footnote{http://trilinos.sandia.gov/} packages BELOS and IFPACK. Some details of the implementation of the Monodomain preconditioning strategy are explained hereafter.
Solution of linear system in (3.8):

- **solver**: Flexible Block-GMRES (with block size set to 1) implemented in BELOS;
- **stopping criterion**: control of 2-norm of the current residual, normalized with respect to the 2-norm of the initial residual
- **tolerance**: set to $10^{-5}$. It is named outer tolerance.

Solution of linear systems in (3.10):

- **solver**: Block-CG (with block size equal to 1), implemented in BELOS;
- **preconditioner**: ILU left preconditioner, implemented in IFPACK;
- **stopping criterion**: control of 2-norm of the current residual, normalized with respect to the 2-norm of the initial residual
- **tolerance**: will be discussed afterwards. It is named inner tolerance.

Further details on the code implementation are discussed in Section 5.2. All the computations are carried out on a workstation equipped with a 2.2 GHz AMD Dual-Core Opteron processor and 4 GB RAM.

To compare the performances of the proposed preconditioning strategy with a reference solver, we consider at first two possible alternative reference solvers, namely the GMRES method applied to the non-symmetric formulation (3.8) that originates our preconditioner and the CG method applied to the symmetric Bidomain system, both preconditioned with an ILU factorization. The latter at each time $t_{n+1}$ reads

$$B_S u_S^{n+1} = f_S^n$$

(3.21)

where

$$B_S = \begin{bmatrix} B_{Sii} & B_{Si}^e & B_{Sei} \\ B_{S}^e & B_{Se}^e & B_{S}^{ee} \end{bmatrix}, \quad u_S = \begin{bmatrix} u_i \\ u_e \end{bmatrix}, \quad f_S = \begin{bmatrix} f_i \\ f_e \end{bmatrix}$$

and

$$B_{Sii}^S = \frac{M}{\delta t} + K_i, \quad B_{Si}^e = B_{Sei}^S = -\frac{M}{\delta t}, \quad B_{Se}^{ee} = \frac{M}{\delta t} + K_e.$$

Entries of vectors $u_i$ and $u_e$ are the nodal values of $u_i^h$ and $u_e^h$, while vectors $f_i$ and $f_e$ represent the discretization of the forcing terms in the two equations. For the solution of system (3.21) we used:

- **solver**: Block-CG (with block size equal to 1), implemented in BELOS;
- **preconditioner**: ILU left preconditioner, implemented in IFPACK;
- **stopping criterion**: control of 2-norm of the current residual, normalized with respect to the 2-norm of the initial residual.
• tolerance: $10^{-5}$.

The ILU-GMRES method is used for the same formulation to which our preconditioner is applied so it is the natural candidate for our comparisons. On the other hand, ILU-CG method exploits the symmetry of the original formulation of the problem, so it is supposed to be more effective. Preliminary computations on a truncated ellipsoidal geometry reported in Table 3.1 show that the two approaches feature similar performances. The ionic model used in this test is the Rogers-McCulloch, and the simulations are run for 50 ms. In the first column we report the number of nodes of the computational meshes used in the simulations. In the second and fourth columns we report the average execution time for the solution of the Bidomain linear system with ILU-CG and ILU-GMRES respectively. The average is computed over all the time iterations of the simulations, with the exception of the first one, which is the most expensive one, since the ILU factorizations are computed at this stage. In the third and fifth columns we show the average iteration counts for the solution of the Bidomain linear systems obtained with the mentioned solvers. In this case the average is computed over all the time iterations of the simulations.

<table>
<thead>
<tr>
<th># nodes</th>
<th>ILU-CG</th>
<th>ILU-GMRES</th>
</tr>
</thead>
<tbody>
<tr>
<td>29560</td>
<td>4.32848</td>
<td>4.91394</td>
</tr>
<tr>
<td>62566</td>
<td>13.4555</td>
<td>17.1016</td>
</tr>
<tr>
<td>172878</td>
<td>61.561</td>
<td>88.8527</td>
</tr>
</tbody>
</table>

Table 3.1: Comparison of the performances of ILU-CG applied to the symmetric formulation (3.21) and ILU-GMRES applied to the non-symmetric formulation (3.8) of the Bidomain linear system coupled with the Rogers-McCulloch ionic model, for different mesh sizes and an ellipsoidal geometry: average execution time per time step, and average iteration counts per time step for the solution of the Bidomain linear systems with the ILU-CG and ILU-GMRES solver, respectively.

Since the performances of the two alternative solvers are comparable, in the sequel, we compare our results with the ILU-CG solver, which is based on the most popular formulation of the problem.

As shown in [60], the level of sparsity used to compute the ILU preconditioner (both for the symmetric Bidomain solver and for the Monodomain blocks solver in the Monodomain preconditioner strategy) influences the number of CG iterations to fulfill the required tolerance, but the computational time is substantially independent of it. In BELOS ILU preconditioner, the level of sparsity is driven by the level of fill (see [136]) and lower is the level, closer the pattern of the L and U factors is to the pattern of the original matrix.

The significant reduction in the number of iterations of ILU-CG with higher level of fill, could be an advantage in parallel implementations and will be investigated in a future
Numerical methods for electrocardiology

Figure 3.4: Left: truncated ellipsoidal geometry representing an idealized left ventricle. Right: real ventricular geometry reconstructed from SPECT images. White arrows represent myocardial fiber orientation used in our numerical simulations (see [33] for their analytical description).

development of this work. In the remainder of this Chapter, the level of fill is set to 1.

Influence of the inner tolerance and of \( \lambda \) This set of numerical experiments aims at investigating the robustness of the preconditioner with respect to the accuracy in the solution of systems (3.10). We performed numerical simulations over an idealized ventricular geometry represented by the truncated ellipsoid reported in Figure 3.4 (left).

We set \( \lambda = 1.3 \), and the simulation runs for 50 ms with the Luo-Rudy phase I model. We solve here systems (3.10) with a tolerance \( tol. = 10^{-5} \), which is the same tolerance used as a stopping criterion in the outer iterations. Then we solve the problem with a coarse tolerance \( tol. = 0.12 \) for solving systems (3.10), which is the result of a fine tuning for finding a trade-off between the number of outer iterations and CPU time to solve (3.10).

In Table 3.2 we report the average CPU time and the average FGMRES iteration counts over the entire simulation with different mesh sizes. The two solutions of the Bidomain systems are computed up to the fulfillment of the same outer tolerance on the residual. Table 3.2 highlights the relevant CPU time reduction with the use of a coarse inner tolerance, while the outer iteration counts are almost insensitive to the different accuracy required in the solution of the preconditioned systems.

A test comparing three different inner tolerances (0.12, 0.01 and \( 10^{-5} \)) on a wide range of mesh size, from 22470 to 677000 nodes for the real geometry introduced above has been performed as well. We report in Figure 3.5 the results obtained. As expected, as the inner tolerance decreases the average CPU time increases and the average number of iterations decreases. However, the iteration count reduction is small in comparison with the increase of CPU time. This results suggest to use an inner tolerance of 0.12 in performing the subsequent tests.
3 Numerical methods for electrocardiology

<table>
<thead>
<tr>
<th></th>
<th>tol. = $10^{-5}$</th>
<th></th>
<th>tol. = 0.12</th>
</tr>
</thead>
<tbody>
<tr>
<td>nodes</td>
<td>time</td>
<td>iter</td>
<td>time</td>
</tr>
<tr>
<td>29560</td>
<td>3.89158</td>
<td>2.988</td>
<td>1.08365</td>
</tr>
<tr>
<td>62566</td>
<td>10.6337</td>
<td>3</td>
<td>3.82078</td>
</tr>
<tr>
<td>172878</td>
<td>53.0898</td>
<td>3.922</td>
<td>15.4344</td>
</tr>
</tbody>
</table>

Table 3.2: Comparison of the performances of the Monodomain preconditioner with a fine vs coarse inner tolerance.

Figure 3.5: Comparison of the performances of the preconditioner with different values of inner tolerance on different meshes. Inner tolerance is set to $10^{-5}$ (dotted line), 0.01 (dashed line) and 0.12 (solid line). Left: average CPU time. Right: average iteration counts.

In order to choose the value of $\lambda$ used in our numerical tests, we have performed some computations on real geometry meshes with different size, trying different values of $\lambda$ within the range $[\lambda_m, \lambda_M]$. In particular we selected $\lambda = 0.6667, 0.9833, 1.3, 2.7934, 4.2868$. This list contains the minimum and maximum values of $\lambda$, $\lambda_m, \lambda_M$, and the value $\lambda = 1.3$ found empirically to minimize the CPU time for the simulation. We also added to the list the midpoint between $\lambda_m$ and 1.3 and the midpoint between 1.3 and $\lambda_M$.

We compared the performances obtained using three different meshes with 22470, 276578, 677000 nodes, with respect to the average CPU time and the average iteration counts. Results are reported in Figure 3.6. In a second set of simulations we also tested the value $\lambda = \lambda_*$ found with the minimization of the condition number of the preconditioned operator $P$. We compared the results obtained, in terms of iteration count and CPU time, with the ones obtained with $\lambda = 1.3$, obtaining equivalent results. However we do not report these new results in Figure 3.6 since they are acquired with a different set up of the software libraries employed and are therefore not directly comparable with those already presented.

Screenshots of the solution are reported in Figure 3.7.

The average number of iterations is quite insensitive to the choice of $\lambda$ for all the mesh...
3 Numerical methods for electrocardiology

Figure 3.6: Comparison of the performances of the Monodomain preconditioner with different values of $\lambda$ on meshes with different size. Dashed line: average number of iterations. Solid line: average computational time. Results are obtained with a real ventricle mesh with 22470 (left), 276578 (center) and 677000 (right) nodes.

Figure 3.7: Screenshots of the action potential propagation (in mV) at t=70ms (left) and t=400ms (right), computed on a real left ventricular geometry.

tested, while the average computational time features an “optimal” value that depends on the mesh size. However, we can observe that for very different mesh sizes, the best value of $\lambda$ lays around the interval $[1,1.5]$. Therefore we suggest to select $\lambda$ in this range.

Heartbeat simulation In this test we analyze the effectiveness of the Monodomain preconditioner from the depolarization to the repolarization (500ms) of the ventricle tissue in one cardiac cycle on a fine grid. In particular we choose $h = 0.02$ cm on a computational domain given by a slab geometry of size 1x1x0.2 cm, that can be handled on a single processor computer. The resulting tetrahedral grid counts 208,848 vertices.

The Bidomain system is coupled with the Luo-Rudy Phase I model. We set again $\lambda = 1.3$, and we solve the systems in (3.10) with an inner tolerance $tol. = 0.12$. We plot in Figure 3.8 (top) the evolution of the iteration counts for both the ILU preconditioned

61
problem and the Monodomain preconditioned problem (denoted $M_{\text{Prec}}$). ILU preconditioner simulation shows a remarkable variation in the iteration counts at the beginning and at the end of the simulation, as already observed in [115]. Correspondingly, CPU time at the beginning and at the end of simulation is increased. This phenomenon at the moment does not have a clear justification. A possible heuristic explanation is that it is a consequence of the discretization errors due to large variations in the ionic variables, in correspondence to the opening and closing of gating channels, occurring at the beginning and the end of APD. These errors could be amplified by the ill conditioning of the Bidomain system. On the other hand the Monodomain preconditioner has a fairly constant performance along the whole simulation in terms of number of iterations per time step, which is remarkably smaller than in the ILU preconditioner case. CPU time plot (Figure 3.8, bottom) shows that, also for our preconditioner, the CPU time slightly increases, as should be expected because of the large variations of the ionic current. This effect is however less evident for our preconditioner versus the ILU one. Figure 3.9 shows transmembrane and extracellular potentials computed with ILU preconditioner (dashed line) and Monodomain preconditioner (solid line). Computed solutions are clearly the same.

Influence of the mesh size  In this test we run the Bidomain simulations on the truncated ellipsoid for 50 ms. We set again $\lambda = 1.3$, while the tolerance for systems (3.10) is $\text{tol.} = 0.12$. We compare the iteration counts and the execution time of the Bidomain linear system solution, for both solvers. In particular, since the first time step is by far the most expensive, as the ILU factorizations are carried out at this stage, we separate the contribution of the execution time of the first time step from the average execution time over the remaining time steps. The average iteration counts of the conjugate gradient algorithm are computed on the overall simulation. Results in Table 3.3 refer to Rogers-McCulloch model, while results in Table 3.4 refer to Luo-Rudy phase I model. The iteration counts of the Monodomain preconditioner appear to be essentially insensitive to the mesh size for both ionic models. Execution time of the preconditioned system remains significantly lower than the one of the symmetric Bidomain problem (see Table 3.5), the differences becoming more pronounced when we use finer meshes. The difference is particularly evident in the execution time of the first time step when the incomplete LU factorization is carried out. This feature is likely relevant when the LU factorization needs to be frequently repeated during the simulations, like, for instance, when the movement of the cardiac tissue is included in the model. It is worth noticing that in order to accurately describe the sharp fronts of potentials and the propagation velocity, finer mesh and smaller time steps should be used. Mesh sizes used in this work are basically limited by the use of serial architectures. In this respect, our preconditioner not only reduces the CPU time in comparison with the ILU-CG, but demands for less storage resources, since the ILU factorization is performed on smaller matrices. On the other hand, the optimality of the Monodomain preconditioner is a promising feature in view of parallel implementations that allow the use of finer meshes.
3 Numerical methods for electrocardiology

<table>
<thead>
<tr>
<th># nodes</th>
<th>1st step</th>
<th>time</th>
<th>iter</th>
<th>1st step</th>
<th>time</th>
<th>iter</th>
</tr>
</thead>
<tbody>
<tr>
<td>12586</td>
<td>9.66</td>
<td>1.31869</td>
<td>22.75</td>
<td>2.77</td>
<td>0.885859</td>
<td>5.03</td>
</tr>
<tr>
<td>29560</td>
<td>25.53</td>
<td>4.32848</td>
<td>29.85</td>
<td>7.53</td>
<td>2.87768</td>
<td>6</td>
</tr>
<tr>
<td>62566</td>
<td>61.34</td>
<td>13.4555</td>
<td>37.32</td>
<td>18.12</td>
<td>7.76394</td>
<td>6.01</td>
</tr>
<tr>
<td>127401</td>
<td>137.03</td>
<td>40.6704</td>
<td>46.29</td>
<td>41.35</td>
<td>20.6259</td>
<td>6.01</td>
</tr>
<tr>
<td>172878</td>
<td>211.91</td>
<td>61.561</td>
<td>48.16</td>
<td>61.28</td>
<td>31.6269</td>
<td>6.02</td>
</tr>
<tr>
<td>508383</td>
<td>1043.32</td>
<td>392.806</td>
<td>78</td>
<td>295.95</td>
<td>181.893</td>
<td>7</td>
</tr>
<tr>
<td>841413</td>
<td>1939.73</td>
<td>779.183</td>
<td>92.56</td>
<td>514.2</td>
<td>329.209</td>
<td>7</td>
</tr>
</tbody>
</table>

Table 3.3: Rogers-McCulloch model: execution time (in s) for the first time step, average execution time (in s) per time step (excluding the first one), and average iteration counts per time step. Columns 2-4: symmetric Bidomain with ILU preconditioned CG. Columns 5-7: non-symmetric Bidomain with Monodomain preconditioner.

<table>
<thead>
<tr>
<th># nodes</th>
<th>1st step</th>
<th>time</th>
<th>iter</th>
<th>1st step</th>
<th>time</th>
<th>iter</th>
</tr>
</thead>
<tbody>
<tr>
<td>12586</td>
<td>9.27</td>
<td>0.659178</td>
<td>10.358</td>
<td>2.37</td>
<td>0.425992</td>
<td>3.006</td>
</tr>
<tr>
<td>29560</td>
<td>23.42</td>
<td>1.80633</td>
<td>11.268</td>
<td>5.96</td>
<td>1.08365</td>
<td>3.068</td>
</tr>
<tr>
<td>62566</td>
<td>53.49</td>
<td>4.8288</td>
<td>12.136</td>
<td>13.58</td>
<td>3.82078</td>
<td>3.972</td>
</tr>
<tr>
<td>127401</td>
<td>118.06</td>
<td>13.779</td>
<td>14.478</td>
<td>33.47</td>
<td>9.28469</td>
<td>3.992</td>
</tr>
<tr>
<td>172878</td>
<td>170.27</td>
<td>22.0005</td>
<td>16.076</td>
<td>46.55</td>
<td>15.4344</td>
<td>3.99</td>
</tr>
<tr>
<td>508383</td>
<td>767.96</td>
<td>134.46</td>
<td>26.894</td>
<td>198.45</td>
<td>58.1504</td>
<td>4.04</td>
</tr>
<tr>
<td>841413</td>
<td>1509.6</td>
<td>294.201</td>
<td>33.268</td>
<td>333.65</td>
<td>149.989</td>
<td>4.9</td>
</tr>
</tbody>
</table>

Table 3.4: Luo-Rudy phase I model: execution time (in s) for the first time step, average execution time (in s) per time step (excluding the first one), and average iteration counts per time step. Columns 2-4: symmetric Bidomain with ILU preconditioned CG. Columns 5-7: non-symmetric Bidomain with Monodomain preconditioner.
A block Gauss-Seidel preconditioner for the symmetric formulation

The idea of using a block-triangular preconditioner could be applied also to the symmetric formulation (3.21) of the Bidomain problem, by simply dropping the block $B^S_{ie}$ in the preconditioner. Numerical results show that this choice is not effective. As a matter of fact, for a mesh of 29560 nodes (with the Luo-Rudy Phase I ionic model), for instance, the execution time for the first time step is 31.02s, the average iteration count is 87.114, and the average execution time is 24.4803s, showing that this choice is more expensive than both the Monodomain preconditioner based on the non-symmetric formulation and the ILU-CG preconditioner (compare these execution times with those reported in Table 3.4). This can be explained by observing that to drop block $B^S_{ie}$ amounts to neglect a significant part of the Bidomain system (3.21), in particular when $\delta t \to 0$. On the contrary, in the
3 Numerical methods for electrocardiology

Figure 3.9: Time evolutions of transmembrane and extracellular potentials (in mV) at a fixed spatial point in the slab. Solutions obtained with conjugate gradient method and ILU preconditioner (dashed line), and with flexible GMRES and Monodomain preconditioner (solid line).

Table 3.5: Ratio of the CPU times and ratio of iteration counts between conjugate gradient method with ILU preconditioner and flexible GMRES with Monodomain preconditioner. Rogers-McCulloch model (columns 2-4) and Luo-Rudy phase I model (columns 5-7).

<table>
<thead>
<tr>
<th># nodes</th>
<th>ILU / Mprec (RMC)</th>
<th>ILU / Mprec (LR1)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1st step</td>
<td>time</td>
</tr>
<tr>
<td>12586</td>
<td>3.4874</td>
<td>1.4886</td>
</tr>
<tr>
<td>29560</td>
<td>3.3904</td>
<td>1.5042</td>
</tr>
<tr>
<td>62566</td>
<td>3.3852</td>
<td>1.7331</td>
</tr>
<tr>
<td>127401</td>
<td>3.3139</td>
<td>1.9718</td>
</tr>
<tr>
<td>172878</td>
<td>3.4581</td>
<td>1.9465</td>
</tr>
<tr>
<td>508383</td>
<td>3.5253</td>
<td>2.1595</td>
</tr>
<tr>
<td>841413</td>
<td>3.7723</td>
<td>2.3668</td>
</tr>
</tbody>
</table>

non-symmetric formulation (3.8) the effect of dropping block $B_{uc}$ is less relevant since the neglected block is the difference of two terms that can be made “small” for a suitable choice of the Monodomain parameter $\lambda$.

3.3 Model adaptivity

In this part of the thesis, we propose another approach to simulate efficiently the action potential propagation in the heart, that can be easily coupled with the one presented in Section 3.2 to save more computational time. This approach has been presented in [94], where part of the results presented in this Section are introduced. More pre-
3 Numerical methods for electrocardiology

...cisely, inspired by the recent literature on modeling error estimation and adaptation (see e.g. [19, 111, 119, 124, 160]), we combine the Bi- and Mono-domain models in a model adaptivity framework. The basic idea is to confine the (more expensive) Bidomain solution to a small part of the domain at hand, while on its most part we solve the Monodomain equation. In this way, we reduce the computational time, without significantly affecting the reliability of the numerical solution.

The crucial step in this approach is the set up of a modeling error estimator able to identify the region where it is worth solving the Bidomain system. Based on the error estimate we solve a finite element discretization of the hybrid model. We actually solve the Bidomain model on some elements whilst in the most part of the domain we keep on solving the Monodomain system. In Section 3.3.1 we introduce the hybrid model used for the model adaptivity and present the semi-discretization of these problems (continuous in space, discrete in time). In Section 3.3.2 we introduce two quantities providing a posteriori upper and lower bounds for the modeling error and investigate their properties. In Section 3.3.3 we describe the algorithm used to achieve model adaptivity and in Section 3.3.4 we show numerical results on real and idealized geometries, proving the good properties of the estimators and the effectiveness of the adaptive strategy. Finally in Section 3.3.5 we draw possible future developments and extensions of the present work.

3.3.1 The Hybridomain model

Since our approach is to combine in the same computation both the Bidomain and Monodomain models, also in this situation we use the non-standard formulations of Monodomain and Bidomain problems introduced in Section 3.2. More precisely, we propose to split the domain $\Omega$ into two parts $\Omega_B$ and $\Omega_M$, being $\Omega_M \cup \Omega_B = \Omega$ and $\Omega_M \cap \Omega_B = \emptyset$ and to solve problem (3.4) on $\Omega_B$ and (3.5) on $\Omega_M$. The hybrid model obtained by combining them has been named Hybridomain and reads:

$$\begin{cases}
\chi C_m \frac{\partial u}{\partial t} - \nabla \cdot \left( \frac{\lambda D_i}{1 + \lambda} \nabla u \right) - \nabla \cdot \left( \frac{\lambda D_i - D_e}{1 + \lambda} \nabla u_e \right) + \chi I_{\text{ion}}(u) = I_{\text{app}} \\
- \nabla \cdot \left[ D_i \nabla u + (D_i + D_e) \nabla u_e \right] = \tilde{I}_{\text{app}}.
\end{cases}$$

(3.22)

where $\mathbb{1}_{\Omega_B}$ is the characteristic function defined in $\Omega$, so that $\mathbb{1}_{\Omega_B}(x, y, z) = 1$ for $(x, y, z) \in \Omega_B$ and $\mathbb{1}_{\Omega_B}(x, y, z) = 0$ elsewhere. In the adaptive strategy we propose, $\Omega_B$ can be updated at each time step, on the basis of the error estimator, as described in Section 3.3.3. We recall that the intra and extra cellular potentials $u_i$ and $u_e$ are defined only up to the same function of time. Again, we will fix such function by requiring that $u_e$ has zero average.

Starting from (2.14), assuming the intra and extra cellular applied currents to be equal and taking $\phi_i = u_i$ and $\phi_e = u_e$, so that $\phi = u$, we obtain

$$\frac{\chi C_m}{2} \int_{\Omega} \frac{\partial u^2}{\partial t} + \|u_i, u_e\|_{E(\Omega)}^2 = -(I_{\text{ion}}(u), u) + (I_{\text{app}}^i, u).$$

(3.23)
The norm $\| \cdot \|_{E(\Omega)}$ is defined in (2.15). Equation (3.23) states the balance of the energy of the considered physical system, the first term on the right hand side being the energy rate by ionic currents and the second term on the righthand side being the energy rate by external currents. Given a specific form of the ionic model, an estimate of the energy rate by ionic currents in terms of the solution $u$ can be deduced.

We consider a semi-implicit first order time advancing scheme, as described in Section [3.1.1] where the terms depending on the ionic currents are taken at the previous time step, so that at each time step the problem is linear. Stability bounds induced by this choice are in general not too restrictive in practice. Let $\delta t$ be the (constant) time step of the discretization. Denote with superscript $n$ the variables computed at time $t^n = t^0 + n\delta t$. We denote with $(u^n_{i,B}, u^n_{e,B}, u^n_B = u^n_{i,B} - u^n_{e,B})$ the solution to (3.4), with $(u^n_{i,M}, u^n_{e,M}, u^n_M = u^n_{i,M} - u^n_{e,M})$ the solution to (3.6) and with $(u^n_H, u^n_e, u^n_B = u^n_H - u^n_e)$ the solution to the Hybridomain problem (3.22). Moving from time step $t^n$ to $t^{n+1}$ the semi-implicit time-discretization of (3.4) reads

$$
\begin{cases}
\chi_m C u^{n+1}_B - u^n_B = \frac{\lambda D_i}{1 + \Lambda} \nabla u^{n+1}_B - \nabla \left( \frac{\lambda D_i - D_e}{1 + \Lambda} \nabla u^{n+1}_B \right) + \\
\chi I_{ion}(u^n_B) = \tilde{I}^{app,n+1} \\
- \nabla \cdot \left[ D_i \nabla u + (D_i + D_e) \nabla u^{n+1}_B \right] = \tilde{I}^{app,n+1}.
\end{cases}
$$

Similarly, for the Hybridomain (3.22), we resort to the following discretization

$$
\begin{cases}
\chi_m C u^{n+1}_H - u^n_H = \frac{\lambda D_i}{1 + \Lambda} \nabla u^{n+1}_H - \nabla \left( \frac{\lambda D_i - D_e}{1 + \Lambda} \nabla u^{n+1}_H \right) + \\
- \nabla \cdot \left[ D_i \nabla u^{n+1} + (D_i + D_e) \nabla u^{n+1}_e \right] = \tilde{I}^{app,n+1} - \chi I_{ion}(u^n).
\end{cases}
$$

Notice the choice $1_{\Omega^n_B}$, which implies that the region where we switch the “Bidomain” term on is estimated upon the solution at the previous time step. A similar discretization is carried out for the Monodomain (3.6) model.

After discretizing (3.24) in space for instance by Lagrange finite elements, we are led to an algebraic system of the form

$$
\begin{bmatrix}
B_{uu} & B_{ue} \\
B_{eu} & B_{ee}
\end{bmatrix}
\begin{bmatrix}
u^{n+1}_B \\
u^{n+1}_e
\end{bmatrix}
= 
\begin{bmatrix}
f^{n+1} \\
g^{n+1}
\end{bmatrix},
$$

being $\mathbf{u}^{n+1}_B$ and $\mathbf{u}^{n+1}_B$ the vectors of nodal values corresponding $u^{n+1}_B$ and $u^{n+1}_e$, respectively. Similarly, the Monodomain problem will read at algebraic level

$$
\begin{bmatrix}
B_{uu} & B_{ue} \\
B_{eu} & B_{ee}
\end{bmatrix}
\begin{bmatrix}
u^{n+1}_M \\
u^{n+1}_M
\end{bmatrix}
= 
\begin{bmatrix}
f^{n+1} \\
g^{n+1}
\end{bmatrix},
$$

In the Hybridomain approach we will assemble the off diagonal matrix $B_{ue}$ only in those elements $K \in \Omega^B$. 

67
3 Numerical methods for electrocardiology

3.3.2 The \textit{a posteriori} estimators

The differences between the Bidomain and Monodomain solutions at the generic time $t^{n+1}$ are denoted by

\[
\begin{align*}
\epsilon_{i,M}^{n,tot} &= u_{i,B}^{n+1} - u_{i,M}^{n+1} \\
\epsilon_{e,M}^{n,tot} &= u_{e,B}^{n+1} - u_{e,M}^{n+1} \\
\epsilon_{u,M}^{n,tot} &= u_{i,B}^{n+1} - u_{M}^{n+1}
\end{align*}
\] (3.26)

while

\[
\begin{align*}
\epsilon_{i,H}^{n,tot} &= u_{i,B}^{n+1} - u_{i,H}^{n+1} \\
\epsilon_{e,H}^{n,tot} &= u_{e,B}^{n+1} - u_{e,H}^{n+1} \\
\epsilon_{u,H}^{n,tot} &= u_{i,B}^{n+1} - u_{H}^{n+1}
\end{align*}
\] (3.27)

denote the differences between the Bidomain and the Hybridomain models. In the sequel we set $D_e \equiv \frac{D_e - \lambda D_i}{1 + \lambda}$.

We split the differences (3.27) (resp. (3.26)) in two components. Let $(\tilde{u}_B^{n+1}, u_{e,B}^{n+1}, \tilde{u}_i^{n+1})$ be the solution of (3.24) from time step $t^n$ to $t^{n+1}$ moving from the Hybridomain (resp. Monodomain) solution at time $t^n$, namely

\[
\begin{cases}
\chi_{C_m} \frac{\tilde{u}_B^{n+1} - \tilde{u}_B^n}{\delta t} - \nabla \cdot \left( \frac{\lambda D_i}{1 + \lambda} \nabla \tilde{u}_B^{n+1} \right) - \nabla \cdot \left( \frac{\lambda D_e - D e}{1 + \lambda} \nabla \tilde{u}_i^{n+1} \right) + \chi I_{ion}(\tilde{u}_B^n) = \tilde{f}_{app}^{n+1} \\
- \nabla \cdot \left[ D_i \nabla \tilde{u}_B^{n+1} + (D_i + D_e) \nabla \tilde{u}_e^{n+1} \right] = \tilde{f}_{app}^{n+1}, \\
\tilde{u}_B^n = u_{i,H}^n, \quad \tilde{u}_e^n = u_{e,H}^n
\end{cases}
\] (3.28)

Then the total error is split as

\[
\begin{align*}
\epsilon_{i,H}^{n+1,tot} &= \epsilon_{i,H}^{n+1} + e_{i,H}^{n,n}, \\
\epsilon_{e,H}^{n+1,tot} &= \epsilon_{e,H}^{n+1} + e_{e,H}^{n,n}, \\
\epsilon_{u,H}^{n+1,tot} &= \epsilon_{u,H}^{n+1} + e_{u,H}^{n,n}
\end{align*}
\]

and similarly for the other components $\epsilon_{u,M}^{n+1,tot}$ and $\epsilon_{e,M}^{n+1,tot}$. The component $(\epsilon_{u,H}^{n+1}, \epsilon_{i,H}^{n+1}, \epsilon_{e,H}^{n+1})$ can be considered the \textit{local contribution} to the error, being the difference introduced at each time step by using the Hybridomain model instead of the Bidomain one, starting from the same solution at time $t^n$. The contribution $(\epsilon_{u,M}^{n+1}, \epsilon_{i,M}^{n+1}, \epsilon_{e,M}^{n+1})$ is a propagated error whose analysis involves the stability of the time discrete Bidomain operator, coupled with the ionic model (see e.g. [16]). Hereafter, we focus on estimating the local error only. Numerical results will show that this achieves an effective control also on the total error.

Definition of the upper bound estimator

Recalling that $\Omega \setminus \Omega_B = \Omega_M$, memberwise subtraction of the equations (3.25) to the corresponding ones of (3.28) yields the error equation

\[
\begin{cases}
\chi_{C_m} \frac{\epsilon_{u,H}^{n+1}}{\delta t} - \nabla \cdot \left( \frac{\lambda D_i}{1 + \lambda} \nabla \epsilon_{u,H}^{n+1} \right) + \nabla \cdot \left( D_e \nabla \epsilon_{e,H}^{n+1} \right) = - \nabla \cdot \left( D_e \chi_{\Omega_M} \nabla \epsilon_{e,H}^{n+1} \right) \\
- \nabla \cdot \left[ D_i \nabla \epsilon_{u,H}^{n+1} + (D_i + D_e) \nabla \epsilon_{e,H}^{n+1} \right] = 0,
\end{cases}
\]

68
which can be written in equivalent form, upon taking a linear combination of the two
equations with coefficients \((1, 1/(1 + \lambda))\) as

\[
\begin{aligned}
\frac{\chi C_m}{\delta t} e_{u,H}^{n+1} - \nabla \cdot \left( D_i \nabla (e_{u,H}^{n+1} + e_{e,H}^{n+1}) \right) &= -\nabla \cdot \left( D_e \chi C M_n \nabla u_{e,H}^{n+1} \right) \\
-\nabla \cdot \left[ D_i \nabla e_{u,H}^{n+1} + (D_i + D_e) \nabla e_{e,H}^{n+1} \right] &= 0.
\end{aligned}
\] (3.29)

In the sequel we drop time index \(n+1\) for the sake of notation. Notice that if in the
previous system we force \(\Omega^n_M = \Omega\) we estimate the errors \(e_{u,M}, e_{i,M}, e_{e,M}\).
Let us multiply the first equation in the previous system by \(e_{i,H}\) and the second by
\(e_{e,H}\), integrate over \(\Omega\) and sum the two equations. We obtain

\[
\int_\Omega \frac{\chi C_m}{\delta t} e_{u,H}^2 + \int_\Omega \nabla e_{u,H}^T D_i \nabla e_{i,H} - \int_{\partial \Omega} n^T D_i \nabla e_{i,H} e_{u,H} + \\
+ \int_\Omega \nabla e_{e,H}^T D_i \nabla e_{u,H} - \int_{\partial \Omega} n^T D_i \nabla e_{u,H} e_{e,H} + \\
+ \int_\Omega \nabla e_{e,H}^T (D_i + D_e) \nabla e_{e,H} - \int_{\partial \Omega} n^T (D_i + D_e) \nabla e_{e,H} e_{e,H} = \\
= \int_{\Omega^n_M} \nabla e_{u,H}^T D_e \nabla u_{e,H} - \int_{\partial \Omega^n_M} n^T D_e \nabla u_{e,H} e_{u,H}. \] (3.30)

We assume that both problems fulfill the same conditions on the boundary \(\partial \Omega\). For
this reason, in the previous equation we drop the integrals on \(\partial \Omega\).

Exploiting \(e_{u,H} = e_{i,H} - e_{e,H}\), we have

\[
\int_\Omega \nabla e_{u,H}^T D_i \nabla e_{i,H} + \int_\Omega \nabla e_{u,H}^T D_i \nabla e_{u,H} + \int_\Omega \nabla e_{e,H}^T (D_i + D_e) \nabla e_{e,H} = \\
= \int_\Omega \nabla e_{i,H}^T D_i \nabla e_{i,H} + \int_\Omega \nabla e_{e,H}^T D_e \nabla e_{e,H} \] (3.31)

so that the variational formulation of the error system reads

\[
\int_\Omega \frac{\chi C_m}{\delta t} e_{u,H}^2 + \int_\Omega \nabla e_{i,H}^T D_i \nabla e_{i,H} + \int_\Omega \nabla e_{e,H}^T D_e \nabla e_{e,H} = \\
= \int_{\Omega^n_M} \nabla e_{i,H}^T D_i \nabla u_{e,H} - \int_{\Omega^n_M} \nabla e_{e,H}^T D_e \nabla u_{e,H}. \] (3.32)

Using standard techniques, since \(D_i\) and \(D_e\) are symmetric and positive definite, we
can manipulate the right hand side of the previous equation as follows

\[
\int_{\Omega^n_M} \nabla e_{i,H}^T D_i \nabla u_{e,H} - \int_{\Omega^n_M} \nabla e_{e,H}^T D_e \nabla u_{e,H} \leq \\
\leq \int_{\Omega^n_M} \left| \nabla e_{i,H}^T D_i^{1/2} D_i^{1/2} D_i \nabla u_{e,H} \right| + \int_{\Omega^n_M} \left| \nabla e_{e,H}^T D_e^{1/2} D_e^{1/2} D_e \nabla u_{e,H} \right| \leq \\
\leq \frac{1}{2} \| e_{i,H}, e_{e,H} \|_E^{2} + \frac{1}{2} \int_{\Omega^n_M} \nabla u_{e,H}^T D_e^T (D_e^{-1} + D_e^{-1}) D_e \nabla u_{e,H}. \] (3.33)
yielding

$$\int_{\Omega} \frac{\chi C_m}{\delta t} e_{a,H}^2 + \frac{1}{2} \|e_{i,H}, e_{e,H}\|_{E(\Omega^I_M)}^2 + \|e_{i,H}, e_{e,H}\|_{E(\Omega^B_M)}^2 \leq \frac{1}{2} \int_{\Omega_M} \nabla u_{e,H}^T D_e^T (D^{-1} - 1) D e \nabla u_{e,H} \equiv \eta^2_{\Omega_M} (u_{e,H}) \quad (3.34)$$

In the sequel we denote

$$|||e_{i,H}, e_{e,H}|||^2 \equiv \int_{\Omega} \chi C_H \delta t (e_{i,H} - e_{e,H}) (v_{i,H} - v_{e,H}) + \int_{\Omega} \nabla v_{i,H}^T D_i \nabla e_{i,H} + \int_{\Omega} \nabla v_{e,H}^T D_e \nabla e_{e,H}$$

which is still a norm, so we can write in short

$$|||e_{i,H}, e_{e,H}||| \leq \eta_{\Omega_M} (u_{e,H}).$$

The quantity $\eta_{\Omega_M} (u_{e,H})$ bounds therefore the “local” (in time) difference between the “template” Bidomain model and the Hybridomain solution currently computed.

We point out that with similar arguments it is possible to prove that the “complementary” estimator

$$\eta^2_{\Omega_B} (u_{e,H}) = \frac{1}{2} \int_{\Omega_B} \nabla u_{e,H}^T D_e^T (D_e^{-1} - D_i^{-1}) D_i \nabla u_{e,H}$$

measures the difference between the Hybridomain solution and the Monodomain one.

**Remark** The total error could be obviously split in different ways. In particular, we could split the total error into a local component obtained by solving a time step of Hybridomain system moving from the “exact” Bidomain solution at the previous time step. This is the classical approach in analyzing Ordinary Differential Equations schemes and falls under the name of **localizing assumption**. Following this splitting, we can perform an analysis similar to the one carried out above, yielding an upper bound for the local error given by $\eta_{\Omega_M} (\pi_{e,H})$ where $\pi_{e,H}$ is the solution of the Hybridomain system computed starting from the Bidomain data at the previous time step. Differently than $u_{e,H}$ in (3.34), $\pi_{e,H}$ is not available in current applications (since it needs to know the Bidomain solution that we do not want to compute actually). For this reason we prefer to consider our error splitting leading to (3.34).

**Definition of the lower bound estimator**

Let us start from the error equation (3.32). Observe that the left hand side can be derived from a bilinear symmetric scalar form, that we denote by

$$< [e_{i,H}, e_{e,H}] \cdot [v_{i,H}, v_{e,H}] >_1 \equiv \int_{\Omega} \frac{\chi C_m}{\delta t} (e_{i,H} - e_{e,H}) (v_{i,H} - v_{e,H}) + \int_{\Omega} \nabla v_{i,H}^T D_i \nabla e_{i,H} + \int_{\Omega} \nabla v_{e,H}^T D_e \nabla e_{e,H}.$$
The associated norm is denoted by
\[ \|\|e_{i,H}, e_{e,H}\|\|^2 = \int_{\Omega} \frac{\chi C_m}{\delta t} e_{u,H}^2 + \int_{\Omega} \nabla e_{i,H}^T D_i \nabla e_{i,H} + \int_{\Omega} \nabla e_{e,H}^T D_e \nabla e_{e,H} \]

Notice the equivalence between the two norms \( \|\cdot\| \) and \( \|\|\| \). It is actually verified by direct inspection that
\[ \frac{1}{2} \|\|e_{i,H}, e_{e,H}\|\|^2 \leq \|\|e_{i,H}, e_{e,H}\|\|^2 \leq \|\|e_{i,H}, e_{e,H}\|\|^2. \] (3.35)

By exploiting the properties of the scalar product, and denoting
\[ F([v_{i,H}, v_{e,H}]) = \int_{\Omega^M} \nabla v_{i,H}^T D_i \nabla u_{e,H} - \int_{\Omega^M} \nabla v_{e,H}^T D_e \nabla u_{e,H}, \]
we have from (3.32) that
\[ \|\|e_{i,H}, e_{e,H}\|\|^2 = \left( \sup_{[v_{i,H}, v_{e,H}] \neq [0,0]} \frac{F([v_{i,H}, v_{e,H}])}{\|\|e_{i,H}, e_{e,H}\|\|_T} \right)^2. \]

Consequently, for any choice of the test functions \([v_{i,H}, v_{e,H}]\), we have
\[ \zeta([v_{i,H}, v_{e,H}]) = \frac{1}{2} \frac{|F([v_{i,H}, v_{e,H}])|}{\|\|e_{i,H}, e_{e,H}\|\|_T} \leq \frac{1}{2} \|\|e_{i,H}, e_{e,H}\|\|_T \leq \|\|e_{i,H}, e_{e,H}\|\|. \] (3.36)

Let us consider the family of test functions of the form \([p u_{e,H}, -u_{e,H}]\). In order to have a sharp lower bound, we look for the parameter \(p_{opt}\) which maximizes \(\zeta\).

Upon differentiating \(\zeta([p u_{e,H}, -u_{e,H}])^2\) with respect to \(p\) and equating it to zero, we obtain that the maximum lower bound corresponds to
\[ p = \frac{\int_{\Omega} \nabla u_{e,H}^T D_e \nabla u_{e,H}}{\int_{\Omega} \nabla u_{e,H}^T D_e \nabla u_{e,H}} \equiv p_{opt}. \]

Notice that \(p_{opt}\) would be equal to \(\lambda\) under Monodomain assumption \(D_e = \lambda D_i\).

In the sequel, we set \(\zeta_{opt} \equiv \zeta([p_{opt} u_{e,H}, -u_{e,H}])\) the lower bound for the error \(\|\|e_{i,H}, e_{e,H}\|\|\), holding:
\[ \left( \frac{\int_{\Omega} \nabla u_{e,H}^T D_e \nabla u_{e,H} + \nabla u_{e,H}^T D_e \nabla u_{e,H}}{\sqrt{2} \|\|p_{opt} u_{e,H}, -u_{e,H}\|\|_T} \right) \leq \|\|e_{i,H}, e_{e,H}\|\|. \] (3.37)

3.3.3 The adaptive algorithm

As we have pointed out previously, in our Hybridomain model the region \(\Omega_B\) where the Bidomain model is active is evaluated on the basis of the estimator. More precisely, we introduce the local error indicator
\[ \eta^2_K(u_{e,H}) = \frac{1}{2} \int_K \nabla u_{e,H}^T D_e^T (D_i^{-1} + D_e^{-1}) D_e \nabla u_{e,H}, \quad K \in T_h \] (3.38)
so that, if $N_M$ is the number of elements in $\Omega_M$ (we drop the time index for the sake of notation)

$$\eta_{\Omega_M}(u_e,H) = \sqrt{\sum_{k=1}^{N_M} \eta_k^2(u_e,H)}.$$  

We impose then a uniform distribution of the error among all elements of the mesh $T_h$, namely

$$\eta_K(u_e,H) \leq \frac{\tau}{\sqrt{N}},$$  

being $N$ the total number of elements and $\tau$ a prescribed tolerance. Observe that this choice guarantees that $\eta_{\Omega_M}(u_e,H) \leq \tau$. More precisely, the refinement algorithm reads:

**case 1** if for $K \in \Omega^n_M$ inequality (3.39) is fulfilled, then $K \in \Omega^{n+1}_M$, else

**case 2** if (3.39) is not fulfilled, $K \in \Omega^{n+1}_B$.

The coarsening strategy is based on the complementary estimator $\eta_{\Omega_B}(u_e,H)$. For $K \in \Omega_B$ we compute $\eta_K(u_e,H)$. Then, for a given fraction $\sigma$ we verify the inequality

$$\eta_K(u_e,H) \geq \frac{\sigma \tau}{\sqrt{N}}.$$  

(3.40)

In our numerical tests we used $\sigma = 1$. The coarsening strategy reads

**case 1** if for $K \in \Omega^n_B$ inequality (3.40) is fulfilled, then $K \in \Omega^{n+1}_B$, else

**case 2** if (3.40) is not fulfilled, $K \in \Omega^{n+1}_M$.

It is worth pointing out that in this adaptive strategy the upper right block of the matrix needs to be reassembled at each time step. As we will see in Section 3.3.4, the adaptive strategy is still faster than the full Bidomain solver.

The adaptive algorithm has been implemented within the finite elements library LifeV [1]. Details on the implementation are reported in Section 5.3. The space discretization chosen is a piecewise linear finite element discretization while the time advancing scheme is described in Section 3.1.1.

The solution at each time step of the Hybridomain model is carried out with the Trilinos linear solver GMRES or Flexible GMRES ([136]), implemented in BELOS package. The system is preconditioned with the Extended Monodomain model, as done in [60], with different values of the inner tolerance. The same strategy has been applied to the Bidomain system, to compare performances and results. The Extended Monodomain system is solved blockwise, using ILU-preconditioned CG solver for each block.

---

13http://trilinos.sandia.gov
Remark The estimator $\eta_M(u, H)$ and the lower bound $\zeta_{opt}(u, H)$ provide bounds for the local error. Consistently, in the next Section we investigate efficiency and reliability of the estimator by comparing the Hybridomain solution at time $t^{n+1}$ with the Bidomain solution at the same time step, initialized to the same datum at time $t^n$. In the numerical results presented in Section 3.3.4 we have decided to initialize both the Bidomain and Hybridomain solvers to the Bidomain solution at each time step.

3.3.4 Numerical results

We firstly address the comparison of Bidomain, Monodomain and Hybridomain solvers on the propagation of the action potential in a healthy tissue geometry of the left ventricle. This geometry has been segmented from SPECT images and segmented as described in Section 4.1.2. The computational mesh features 1233256 elements and 199766 nodes. The conductivity of the tissue in this test case is homogeneous non-isotropic due to the presence of the cardiac fibers, as described in Section 2.1. Details on the conductivity parameters and the analytical description of the fibers used for the numerical experiments in Section 3.3.4 can be found in [33]. More precisely, the geometrical parameters of the fibers description have been here adjusted to fit the size of the geometries at hand.

After this first test we simulate the presence of a scar in the ventricle wall and we compare the pattern obtained with the different solvers. In this case the conductivity of the tissue is non-homogeneous, the extra cellular conductivities being amplified, while the intracellular conductivities are reduced, as suggested in [46]. In the first case presented, named artificial scar, the geometry of the heart, the space discretization and the fibers direction used are the same as in the previous test case and the region occupied by the scarred tissue is artificial, meaning with this term that it is an analytical shape (a sphere) not based on experimental evidences. We also present the results of a test performed on a left ventricle geometry obtained by a set of MRI medical images. In this case the images reveal the presence of a scar in the tissue, close to the endocardial surface. From this data it has been possible to extract information on the actual scar geometry and use it in the numerical simulation.

We finally present the results obtained with the model adaptivity algorithm, simulating the propagation of the action potential generated by a stimulus exerted by a pacemaker on a simplified left-ventricular geometry. In particular the chosen geometry is a truncated ellipsoid, already employed in Section 3.2 with an analytical description of the fibers [33]. The computational mesh is composed of 165233 nodes and 876032 elements.

Since the adaptive algorithm is independent of the choice of the ionic model, we analyze the performances of this strategy choosing only one of the two ionic models mentioned in Section 2.1 namely the Rogers-McCulloch one, whose parameters are

---

14This set of images has been provided by Dr. E. Garcia, Emory University, Atlanta, GA, USA.
15Images are courtesy of Dr. J. Oshinski, Emory University, Atlanta, GA, USA.
16Work performed by M. Piccinelli, Emory University, using VMTK software (see Section 4.1.2).
specified in [33]. The parameters of the models used in this Section are reported in A.1 and A.2. The time step is $\delta t = 0.5$ ms and the simulations are carried out for 400 ms. We point out that the transmembrane potential computed with Rogers-Mc Culloch model is shifted by 84 mV with respect to the physiological one, setting 0 mV as rest potential in place of the physiological -84 mV.

**Test cases on a healthy real geometry**

The aim of the first set of simulations is to compare the choice of different thresholds $\tau_{eff} = \tau^2$ (being $\tau$ introduced in (3.39)), with respect to the *effectivity index* of the upper bound estimator defined at each time step as

$$\theta_{up} := \eta_{\Omega_M}(u_e,H) \left/ \|e_i,H, e_e,H\|\right.$$  \hspace{1cm} (3.41)

In Figure 3.10 we show for every timestep the effectivity index (top) and the percentage of Bidomain elements over the total number of elements (bottom), for $\tau_{eff} = 20, 40, 80, 120, 160$. More precisely, since the rest transmembrane potential is zero, the error is dropping to zero in the last phase of the simulation and it forces the effectivity index to grow, even if the Hybridomain solution is identical to the Bidomain one. To filter this effect, in Figure 3.10 we plot $\theta_{up}$ if $\|e_i,H, e_e,H\| > 10^{-3}$ and 0 otherwise.

In Figure 3.11 we plot the average in time of the effectivity index with respect to the chosen threshold. Effectivity index is quite robust with respect to the choice of the threshold, ranging between 4.68 and 4.93. Moreover we identify $\tau_{eff} = 80$ as the threshold value that gives the minimum effectivity index, and therefore the more effective adaptive strategy. In the subsequent simulations we set $\tau_{eff} = 80$.

Figure 3.12 shows the effectivity index of the lower bound estimator, defined as

$$\theta_{low} := \zeta_{opt}(u_e,H) \left/ \|e_i,H, e_e,H\|\right.$$  \hspace{1cm} (3.42)

Figure 3.13 highlights (in red) the distribution in space of the active Bidomain elements (region $\Omega_B$) at three different time steps. Comparing the activation pattern with the Bidomain transmembrane potential pattern we stress that the adaptive strategy, based on the estimator $\eta_{\Omega_M}(u_e,H)$, successfully activates the Bidomain model in the area involved by the propagating front. This confirms the reliability of the a priori error estimator.

Let us now calculate the effectiveness of the adaptive Hybridomain model both in reducing the error with respect to the simplified Monodomain model and in reducing the computational time with respect to the complete Bidomain model.

In Figure 3.14 we compare the norm of the difference $e_{tot}^{u,M}$ with the norm of $e_{tot}^{u,H}$. This comparison clearly shows that solving the Hybridomain in place of the Monodomain produces a solution much closer to the Bidomain one both in terms of the $H^1$ and $L^2$ errors.
In Table 3.6 we compare the computational effort required for solving the Bidomain system and the Hybridomain system. In particular we report the average CPU time (computed over all the time steps of the simulation) and the number of iterations required by the iterative algorithm GMRES to converge. As mentioned in Section 3.3.3 both systems are preconditioned with the Extended Monodomain, and in this test case the preconditioner system is solved with the Conjugate Gradient method, up to the fulfillment of an inner tolerance $tol = 10^{-5}$. In the first row of the table we report the gain in iteration count

$$g_{it} = 100 \frac{(# \text{ Bidomain iterations}) - (# \text{ Hybridomain iterations})}{(# \text{ Bidomain iterations})}.$$ 

In the second row we show the gain in CPU time for the solution of each system in the time advancing scheme, computed as

$$g_{time} = 100 \frac{(CPU \text{ Bidomain}) - (CPU \text{ Hybridomain})}{(CPU \text{ Bidomain})}.$$
Figure 3.11: Average effectivity index for different values of the threshold $\tau_{eff}$. The minimum average is reached for $\tau_{eff} = 80$.

Figure 3.12: Effectivity index $\theta_{low}$ and percentage of active Bidomain elements for $\tau_{eff} = 80$.

When considering that solving the Hybridomain system requires to re-assemble the upper right block of the matrix, the gain is defined as

$$g_{time, net} = 100 \frac{(CPU \ Bidomain)-(CPU \ Hybridomain + CPU \ assembling)}{(CPU \ Bidomain)}.$$ 

This is reported in the last row of the Table. The adaptive strategy we are proposing, combined with the model preconditioning proposed in [60], allows therefore to save more than 30% of CPU time with respect to solving the Bidomain and with an average error per time step (with respect to the Bidomain model) smaller than 2.9%, compared to the 16% average error of the Monodomain solver.
Figure 3.13: Top: Bidomain elements activated in $\Omega$ ($\Omega_B$ shown in red) for $t = 40, 80, 200$ ms. Bottom: Bidomain transmembrane potential (in mV) at $t = 40, 80, 200$ ms.

<table>
<thead>
<tr>
<th></th>
<th>Gain (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$g_{\text{it}}$</td>
<td>28.6</td>
</tr>
<tr>
<td>$g_{\text{time}}$</td>
<td>35.0</td>
</tr>
<tr>
<td>$g_{\text{time, net}}$</td>
<td>33.4</td>
</tr>
</tbody>
</table>

Table 3.6: Percentage gain in CPU effort using the adaptive strategy with respect to solving the Bidomain system. We report in the first row the average gain in iteration count; in the second row the average gain in CPU time required for the solution of the linear system; in the third row the average net gain in CPU time, considering the assembling time required by the Hybridomain.

**Remark** We point out that if we use a coarser inner tolerance for CG method when solving the preconditioner system, as suggested in [60], (Flexible GMRES solver), the computational time required to solve both the Hybridomain and the Bidomain decreases and the difference between them becomes less evident. In particular in this test case
3 Numerical methods for electrocardiology

Figure 3.14: $H^1(\Omega)$ norm (top) and $L^2(\Omega)$ norm (bottom) of $e_{u,M}$ and $e_{u,H}$ at each time step of the simulation.

\( g_{it} = 12.5\% \), \( g_{time} = 10.9\% \) and \( g_{time, net} = 6.2\% \). Effectiveness of adaptivity is less evident in this case. More sophisticated coupling strategies that could avoid to resort to the Extended Monodomain model will allow a more relevant computational time reduction (see [61]) and will be investigated as a future development of the present work.

**Condition number**

Let $B$, $M$ and $H$ be the matrices obtained after the discretization (in time and space) of the Bidomain, (Extended) Monodomain and Hybridomain models at a given time step. We analyze the condition number of the matrices $M^{-1}B$ and $M^{-1}H$ obtained by preconditioning the Bidomain system and the Hybridomain system with the Extended Monodomain matrix. Two different Computational meshes are considered. We report in Table 3.7 the condition number associated. For all the mesh sizes
3 Numerical methods for electrocardiology

<table>
<thead>
<tr>
<th># mesh nodes</th>
<th>$K(M^{-1}B)$</th>
<th>$K(M^{-1}H)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>12437</td>
<td>8.13e+01</td>
<td>5.47e+01</td>
</tr>
<tr>
<td>52953</td>
<td>7.39e+01</td>
<td>6.04e+01</td>
</tr>
</tbody>
</table>

Table 3.7: Condition number of the preconditioned Bidomain ($2^{nd}$) column and Hybridomain ($3^{rd}$) matrices. In the first column we report the number of nodes of the computational mesh used. The Hybridomain matrix corresponds to $t = 15$ ms. From all the Extended Monodomain, Bidomain and Hybrid matrices the singularity has been removed by enforcing, in an algebraic way, the zero-average of the extracellular potential.

Test cases on real geometries in presence of scars

In the first part of this test case we impose an artificial scar on the ventricle wall tissue, on the intersection between the previously described SPECT geometry and a sphere centered in $(0.8,-0.3,0)$ cm and with 1 cm radius, as shown in Figure 3.15. For this test case we run a Bidomain, a Hybridomain and a Monodomain simulation and we compare the transmembrane potential patterns we obtain. In Figure 3.16 we can see that the modeling error estimator activates the Bidomain model in the scar region, even if the propagating front is far from it. This behavior could be useful if the scar region

Figure 3.15: Artificial scar on the wall of the SPECT reconstructed left ventricle geometry in use.
needs to be studied and analyzed in a more accurate way than the rest of the cardiac tissue, during the whole heart beat.

![Bidomain activation and transmembrane potential](image)

Figure 3.16: Left: Bidomain activation ($\Omega_B$ highlighted in red); Right: Bidomain transmembrane potential (in mV) at $t = 40$ ms.

It is also evident from Figure 3.17 that the propagating front predicted by the Hybridomain model near the scar is more similar to the Bidomain front than the Monodomain solution.

We then perform the same kind of test on a different ventricle geometry, taking into account the presence of an image-based scar geometry, depicted in Figure 3.18. The computational domain for the simulation is coloured in grey, while the scar is highlighted in red. For this test case we run a Bidomain and a Hybridomain simulation to compare the transmembrane potential patterns obtained and to show the activation pattern. Some screenshots of the results are reported in Figures 3.19 and 3.20. The activation pattern confirms that the adaptive algorithm is able to track in time the transmembrane potential depolarization (Figure 3.19, $t = 80, 120$ ms) and repolarization (Figure 3.20, $t = 200$ ms) front and that the region of the scar remains active during the simulation.

### Pacemaker stimulation

The goal of this test case is to simulate the propagation of the action potential, when it is initiated by the presence of a pacemaker. This test case is borrowed from [61] and we refer to this paper for omitted details on the set up of the numerical experiment. Another example of simulation of the action potential in presence of a pacemaker can be found in [87].

This kind of test has been chosen because it represents a situation in which the Monodomain results differ from the Bidomain ones, since the propagation is initiated by
3 Numerical methods for electrocardiology

a variation of the extracellular potential, which is not described by the Monodomain model. The pacemaker is simulated by the presence of two point sources outside the ellipsoid, the cathode and the anode, located in \( x_c = (2.75, 0, 0) \) and \( x_a = (3, 0, 0) \) respectively. The distance between the cathode and the epicardial wall is 0.5 mm. We assume the electrodes to be immersed in an isotropic medium of conductance \( g \), representing the blood. The electrical potential \( \varphi \) generated by these sources can be computed analytically, obtaining

\[
\varphi = \frac{I_{stim}}{4\pi g} \left( \frac{1}{|x - x_c|} - \frac{1}{|x - x_a|} \right). \tag{3.43}
\]

The boundary conditions of the Bidomain problem (2.12) need to be modified, to take into account the presence of the external pacemaker. In particular in this case we prescribe:

\[
\begin{align*}
n^T D_i \nabla u + \nabla u_e &= 0 \\
n^T D_e \nabla u_e &= n^T g \nabla \varphi = -\frac{I_{stim}}{4\pi} n^T \left( \frac{x - x_c}{|x - x_c|^3} - \frac{x - x_a}{|x - x_a|^3} \right). \tag{3.44}
\end{align*}
\]

The derivation of the boundary conditions for the Extended Monodomain and the Hybridomain problems is straightforward from (3.44).

The results obtained in this test are presented in Figures 3.21 and 3.22. As expected, the differences between the pattern produced by Monodomain model and the one predicted by the Bidomain are remarkable. Hence in this test case the Monodomain model is a bad approximation of the Bidomain one. On the other hand the Hybridomain pattern is very close to the Bidomain one, confirming the validity of this approach. The activation of the Bidomain region, shown in the upper left quadrant of each set of four screenshots, follows again the propagating front.

3.3.5 Perspectives: Domain Decomposition coupled with Model Adaptivity

In this part of the thesis we have introduced a model adaptivity strategy for coupling Bidomain and Monodomain models in electrocardiology, in the form of a hybrid system, called Hybridomain. In this model, we couple a non symmetric formulation of the Bidomain system and an extended version of the Monodomain one, so that the Hybridomain model is obtained just switching on or off a term locally (i.e. elementwise in the finite element discretization). The region where we activate the “Bidomain term” is selected by using a modeling error estimator introduced here. Numerical results testify the effectivity of the estimator and of the adaptive approach, since it provides a reliable image of the critical regions of potential propagation in the heart, both in the healthy and pathological cases, reducing the computational cost with respect to the standard Bidomain.

There are some limitations that prevent this model adaptive solver to be more effective. In particular, the need of resorting to an Extended Monodomain formulation makes the reduced model still more expensive than the pure Monodomain problem. Coupling
between Bidomain and pure Monodomain problems is however non trivial, for the different nature of the two problems and ad hoc interface conditions need to be devised. This topic has been addressed in [61], where the authors investigate appropriate interface matching conditions for the coupling of Bidomain and “genuine” Monodomain on non-overlapping domains. In this case however the partition of $\Omega$ in $\Omega_M \cup \Omega_B$ is chosen a priori. The natural development we are studying, in collaboration with the authors of [61], is to couple their domain decomposition approach with the adaptive selection of the partition of $\Omega$, according to the computed error estimator.
Figure 3.17: Comparison among the Bidomain, Hybridomain and Monodomain solutions on a scarred ventricle at $t = 90$ ms. On the 1st row we report the Bidomain activation ($\Omega_B$ highlighted in red) on the left and the Hybridomain transmembrane potential on the right; on the 2nd we show the Monodomain transmembrane potential on the left and the Bidomain transmembrane potential on the right. The considered measurement unit for the potential is mV.
Figure 3.18: Ventricle geometry (in grey) and scar region where the tissue conductivity has been modified for the simulation (in red).
Figure 3.19: Comparison between the Bidomain and Hybridomain models on a scarred left ventricle geometry. The first two rows refer to time $t = 80$ ms, while the third and the fourth ones refer to $t = 120$ ms. On the first and third row we show the position of the scar (on the left) and the Bidomain activation, where $\Omega_B$ is represented in red (on the right). On the second and fourth row the Bidomain transmembrane potential is depicted on the left and the Hybridomain transmembrane potential on the right. The considered measurement unit for the potential is mV.
Figure 3.20: Comparison between the Bidomain and Hybridomain models on a scarred left ventricle geometry. The first two rows refer to time $t = 160$ ms, while the third and the fourth ones refer to $t = 200$ ms. On the first and third row we show the position of the scar (on the left) and the Bidomain activation, where $\Omega_B$ is represented in red (on the right). On the second and fourth row the Bidomain transmembrane potential is depicted on the left and the Hybridomain transmembrane potential on the right. The considered measurement unit for the potential is mV.
Figure 3.21: Comparison among the Monodomain, Bidomain and Hybridomain models on a simplified left ventricle geometry, in presence of a pacemaker stimulation. The first two rows refer to time $t = 10$ ms, while the third and the fourth ones refer to $t = 30$ ms. On the first and third row we show the Bidomain activation (on the left) and the Hybridomain transmembrane potential (on the right). On the second and fourth row the Monodomain transmembrane potential is depicted on the left and the Bidomain transmembrane potential on the right. The considered measurement unit for the potential is mV.
Figure 3.22: Comparison among the Monodomain, Bidomain and Hybridomain models on a simplified left ventricle geometry, in presence of a pacemaker stimulation. The first two rows refer to time $t = 50$ ms, while the third and the fourth ones refer to $t = 70$ ms. On the first and third row we show the Bidomain activation (on the left) and the Hybridomain transmembrane potential (on the right). On the second and fourth row the Monodomain transmembrane potential is depicted on the left and the Bidomain transmembrane potential on the right. The considered measurement units is mV.
4 Simulation of biophysical phenomena in moving domains based on medical images

Every physical phenomenon occurring in the cardiovascular system (and, more in general, in the whole human body) involves the interaction of different biological tissues, each featuring peculiar mechanical properties. For instance blood flow in arteries and veins interacts mechanically with the vessel tissue. The pressure exerted by the fluid on the wall contributes to generate its displacement and in turn displacement and velocity of the vessel influence the fluid dynamics. The vessel wall itself is a complex structure, made of layers with different mechanical behavior interacting one with another. In the heart, as described in Chapter 1.1, the action potential propagation is directly influenced by the fibers direction which depends on the mechanics of the tissue. Conversely, the contraction is triggered by the electrophysiology. As a result of the mechanical forces involved, the tissue in the cardiovascular system experiences detectable deformation during the cardiac cycle both in physiological and in pathological conditions. These evidences suggest that the effect of the movement of the tissue in which the phenomena of interest take place should not be neglected in order to reproduce a realistic behavior.

The standard approach to simulate the coupling between different mechanisms is to develop a mathematical model for each of them as well as for the coupling conditions describing their interaction, obtaining a formulation that includes all these components. As summarized in [150], this is a challenging problem from the modeling, mathematical and numerical point of view since the arterial or myocardial wall constitutive laws are yet not assessed and it is not trivial to obtain reliable measurements to validate them. Moreover the considered districts or organs are subject to an external load which is unknown.

The most straightforward way to obtain a discretized solution of a coupled system is to solve simultaneously the discretization of each model and the coupling equations, the so called monolithic approach. However, the resulting discretized system of equations is large, and is generally stiff [150], requiring therefore to develop effective preconditioners. Moreover this strategy requires to develop ad hoc software, since it is not possible to use already available solvers for each component of the coupled model. On the other hand, a possibility is to choose partitioned (or segregated) algorithms, that solve each model separately at each time iteration. The coupling conditions are included in each submodel as suitable boundary conditions or specific terms in the equations and can be enforced in a strong or weak coupling. In the former case, at each time step, subiterations
of the solvers are performed, to satisfy the coupling conditions up to a chosen tolerance. In the second case each subproblem is solved once per time step, producing an explicit scheme and ensuring only a loose coupling among the models.

The advantage of the partitioned strategy is its modularity. The coupled solution is computed by using solvers for the subproblems in each subdomain, typically available, instead of defining a new solver for the specific coupling problem. The disadvantage on the other hand is that if an accurate fulfillment of the coupling conditions is requested, subiterations are made necessary, increasing the computational cost by far. We refer to Chapter 9 of [53] for a detailed and up-to-date description of numerical methods for fluid-structure interaction problems and to [103, 104, 106] for some examples on electro-mechanical coupling in the heart.

Regardless of the technique employed to solve a model which couples different phenomena, the resulting system of equations is in general complex, often involving nonlinearities, and its solution requires specific, often expensive, algorithms able to guarantee the proper energy exchange between the equations. Moreover, while mathematical models for fluid dynamics and for electrocardiology are well established in the literature, models for the vessel wall or for myocardium mechanics are less established and they are not easy to be validated in vivo, making it difficult to include patient specific data in the model, as pointed out in Section 2.1.3.

In this Chapter we present an alternative strategy to include the motion of the domain of interest in the simulation of blood flow in arteries and of the propagation of the action potential in the myocardium. The basic idea is to compute at each time step the displacement of the tissue from available individual medical images and to move the computational domain accordingly. In this way, the phenomenon at hand (blood flow dynamics or action potential propagation) can be simulated in a domain whose configuration is known a priori from a real basis.

The main advantage of this strategy is that it maintains a tight link between numerical simulations and patient specific data, including automatically the mechanical features of the structure, with a limited additional computational cost with respect to the fixed geometry case. The coupled model approach requires a detailed mechanical model, whose parameters are estimated on a patient specific basis, with much more intense modellistic and computational effort. Moreover in most cases, invasive techniques would be necessary to obtain the desired information, sometimes influencing the physical phenomenon itself and hence generating non realistic data.

A similar technique has been proposed in [39] where the authors apply the image-based motion approach to intra-cranial aneurysms and present preliminary results. In this case the geometry is assumed to deform only in the normal direction to the surface and no validation is carried out. Recently, in [150], the same kind of general methodology was applied to perform Finite Volume CFD simulations in patient-specific geometries and compare the results with in-vitro experimental data on a so-called cross aortic phantom, obtaining a good agreement.

Our work fits in the general framework introduced by the cited papers, even if each step of the procedure can be carried out exploiting different strategies and technologies. For instance several approaches can be followed to compute the domain motion from the
available data and various methods can be implemented to simulate blood fluid dynamics in moving domains. In this work we also focused on the validation of the technique, proposing a comparison with a more standard strategy available in the literature.

We finally remark that the image-based (IB) motion strategy can also be employed in other engineering fields where the phenomenon of interest, e.g. fluid dynamics, takes place in a moving domain that can be tracked in time through imaging techniques.

In the following sections we describe in details the procedure we adopt to perform computations in imaged-based moving domains. The starting point of this procedure is a set of medical images acquired at different times during a cardiac cycle by proper medical devices. For each acquisition time, the corresponding images can be processed with specific techniques to reconstruct a geometrical model of the biological tissue at hand. A brief introduction on these topics is reported in Section 4.1. The output of this reconstruction step will be a set of geometrical models (of the whole volume or of its enclosing surfaces) which describes the motion of the considered domain during a cardiac cycle. To take into account the motion of the domain in the simulation, it is necessary to track the movement of each point of the computational grid. This tracking process is often called registration and can be accomplished by means of various techniques, addressed in Section 4.2. In Section 4.2.2 we describe how to extend the motion of the surface points to the interior of the domain in a smooth way. This information is then included in the mathematical formulation of the problems already presented in Sections 2.1.2 and 2.2.1 that can then be implemented and simulated in a moving framework. We then show in Section 4.3 the results obtained on a real test case dealing with blood flow in a human aorta and some preliminary results on the action potential propagation in a moving heart, together with the technical difficulties encountered. We also perform in Section 4.4 a comparison between a standard fluid-structure interaction algorithm and our image-based approach. We then indicate how this work can be extended and further developed. Some details regarding the implementation of the method are described in Section 5.5.

4.1 Geometry reconstruction from medical images

In this Section we introduce some techniques that are used in the literature to build a mathematical representation of a patient specific domain.

4.1.1 Acquisition of medical images

Medical imaging devices are nowadays commonly used in clinical practice, to obtain information about the in vivo anatomy of the human body in a non invasive way. Several techniques can be employed for this purpose and the choice of which one to use depends on the anatomical features of the interested domain, on the desired spatial resolution and acquisition time and on the drawbacks for the patient. Typically the output of a medical imaging device consists in a set of 2D images, each having a well defined position and orientation in space. Such sets can be formed by contiguous images that can
be stacked along a spatial direction, thus defining a volume. Due to technical limitations, only certain imaging modalities allow to capture a series of such datasets at several time instants within a cardiac cycle; the resulting multidimensional dataset obtained is referred to as a 4D dataset.

A detailed analysis of the techniques that best perform in reconstructing vessels anatomy can be found in [53], while a description of imaging techniques useful for geometry reconstruction and motion recovery of the heart can be found in [12]. The reconstruction of myocardial anatomy presents further complexity due to its huge motion, urging for small acquisition time, and due to the presence of surrounding tissues that are not easily distinguished from the myocardium.

We briefly introduce here the imaging modalities that have been involved in the present work to build computational geometries, referring to [53, 58], and to the literature cited therein for further details on these techniques and to other imaging modalities.

All the used modalities create a set of 2D images representing thin slices of the region of interest. In the blood vessel case, these slices are captured at different locations along the vessel curvilinear abscissa, with a well defined spatial orientation and position. In the reconstruction of the heart chambers, short-axis slices are usually employed, that is to say sections perpendicular to the apex-base line.

In a mathematical framework, a medical 3D image (or set of 2D images) displays a scalar function $I: \Omega \rightarrow \mathbb{R}$ which represents the intensity of the recorded signal and associates an intensity value to each point in the acquisition volume $\Omega \subset \mathbb{R}^3$. An image can therefore be seen as the sampling on a regular grid of a continuous function.

**Magnetic Resonance Imaging** MRI scanners exploit on the ability of protons in water molecules contained in biological tissues to align and resonate under the action of magnetic fields. The acquisition requires to expose the patient to magnetic fields with different features: a strong and static field is first applied to align protons. Then a brief application of radio-frequency energy makes protons resonating, and the resulting radio-frequency signal decays at rates characteristic to the tissue in which the protons are bound. At the same time a magnetic field gradient is applied, to encode position into the frequency of the signal. By a proper tuning of the magnetic fields involved, it is possible to achieve a good contrast for the different components of soft tissues.

MRI does not involve the use of ionizing radiation and is therefore considered safer for the patient with respect to other imaging modalities involving x-rays. On the other hand MRI acquisitions cannot be performed on subjects with metallic or electronic implants, such as pace-makers, due to the use of strong magnetic fields.

Since the strength of the signal captured during an MRI acquisition is proportional to the volume element (voxel) that is emitting it, very high spatial resolutions are typically not achievable. In general, it is possible to have a higher resolution in each slice, by reducing the number of acquired slices.

By choosing the proper timing and energy of the applied magnetic field, or by injecting intravenously a paramagnetic contrast agent, it is possible to enhance the contrast
Simulating biophysical phenomena in moving domains based on medical images

For the analysis of the cardiac motion, specific MRI-based acquisition techniques, called cineMRI, can be used, able to acquire about 20 so-called cine frames per cardiac cycle. The best spatial and temporal resolution are achieved by collecting data over multiple, consecutive heartbeats (typically from 5 to 10), in a breathhold regime. During each heartbeat, blocks of data are acquired and organized with respect to electrocardiogram timing, which defines the separate phases or frames of the cardiac cycle. After the acquisition, data from a given phase, collected from the multiple heartbeats, are combined to form the complete image of the particular cine frame.

**Computed Tomography**

Computed tomography (CT) is a technique based on the recording of the projection of x-ray beams through the body and onto a radiographic film, fluoroscopic screen, or digital detector. Differential absorption of x-rays by the various body structures produces contrast in the resulting multislice 2D images, though it is usually only possible to discriminate among bone, soft tissue and air. It can be employed to highlight the vascular lumen anatomy at the expense of the wall and surrounding tissue (angiography) or to evaluate both cardiac structure and function, and in these cases it typically requires the introduction of a radio-opaque dye into the blood stream. In CT angiographies the contrast agent is injected in the venous system instead of intra-artery, to avoid the risk of emboli, associated on the other hand with projection angiography.

The x-ray sources are incorporated in a ring, together with opposing detectors, and this ring is rotated rapidly around the patient, producing the projections from multiple beams, which are then reconstructed into an image. By moving the patient axially through the ring scanner during this process, volumetric images can be reconstructed. If the scanner is provided with multiple rings (as recent technology allows) a high spatial resolution (less than a millimeter) can be achieved also in the axial direction. The acquisition can be synchronized with the heart beat, providing high resolution time resolved imaging, that can be used for the analysis of rapidly moving structures.

This modality provides an excellent spatial resolution and allows to record the motion of the desired tissues, but it requires to use iodinated contrast agents to enhance the contrast between blood and tissue. The main drawback is therefore that the patient must be exposed to a relatively large dose of ionizing radiation that can be harmful to her/his health.

**Single Photon Emission Computed Tomography (SPECT)**

Single Photon Emission Computed Tomography (SPECT) is a technique used to evaluate the myocardial perfusion and the ventricular function. It is performed using a scintillation camera that is able to record images of the gamma ray emissions from organs containing a radiopharmaceutical agent that distributes in the heart in proportion to regional myocardial perfusion.

In this technique the tracer compound is intravenously injected and scintillation camera detectors rotate 180 degrees around the patient in a semicircular or elliptical fashion, collecting a series of planar projection images at regular angular intervals. The three-
4 Simulation of biophysical phenomena in moving domains based on medical images

dimensional (3D) distribution of radioactivity in the myocardium is then reconstructed from the 2D projections, and the resulting data are displayed in series of slices in the short axis, vertical long axis, and horizontal long axis orientations.

The projection images can be recorded at several instants (usually 8 or 16) during the cardiac cycle, synchronizing the acquisition with an electrocardiogram. In this case the technique is called Gated SPECT.

The disadvantage of SPECT for geometry reconstruction purposes is the poor spatial resolution that even decreases as the distance of the imaged object to the scintillation detectors increases. The contrast on the other hand is higher than, for instance, in MRI, since the biological structures surrounding the heart are not visible in this kind of images. The segmentation process is therefore easier but there is no reference system in the image to deduce the torsion of the myocardium with respect to the acquisition axis.

4.1.2 Segmentation and geometry reconstruction

The simulation of biological processes in patient specific domains requires to build a proper representation of the geometry of interest from the intensity function recorded by the imaging process. To accomplish this task, an algorithm able to detect the boundaries that separate the tissue of interest from the surrounding structures should be used, to obtain a mathematical description of the surface (or curve in 2D) that defines the domain of interest.

The process of partitioning a medical image (or a digital image, in general) into multiple sets of pixels (or segments), corresponding to the objects represented in the image and to the background, is called image segmentation. At the end of the process, pixels belonging to the same set will have similar values relatively to a chosen property, generally the intensity function $I$.

It is worth mentioning that, depending on the quality of the acquired images, in some cases it can be useful to preprocess the raw data with specific techniques able to reduce the noise or to enhance the contrast in the intensity function, ahead of the application of a segmentation algorithm. We refer to [69] and to Chapter 4 of [53] for a review on this topic.

There are different segmentation approaches able to extract a mathematical representation of the chosen geometry. Some of them are based on the intervention of a trained operator, other are automatic or semi-automatic, requiring only a few inputs by the operator. An effective automatic technique would be of course desirable, since it would not require operator time and the results would be repeatable. However, often the presence of artifacts in the medical images can be more easily detected by a human operator and, also in low contrast regions of the images, the knowledge of the anatomy can help the operator in performing a correct segmentation.

Among all the available techniques, we mention the class of thresholding methods, the region growing methods and the front propagation techniques which are often used to segment objects for cardiovascular modeling and we will address more in detail the deformable models approach, since it has been used to set up the geometry models used
in the present work. Thresholding methods attempt to determine an intensity value, called the threshold, which separates the desired segments. The segmentation is then achieved by grouping all pixels with intensities greater than the threshold into one class and all other pixels into another class [22]. Region growing methods generate selective segmentations starting from a pre-segmented region (or a single pixel) and iteratively add neighbouring pixels to the region if they satisfy specified homogeneity criteria. The procedure stops when no more pixels can be added to the region. See [120] for advantages and disadvantages of the methods in this class. Front propagation techniques have the goal to track the propagation of a wavefront from a seed point over the image, until it reaches the curve separating two segments. The velocity of the wave depends on image features, being typically lower in regions where the image intensity shows high variability and higher where the image is uniform. The popular fast marching method belongs to this class and can also be used as initialization for more sophisticated ones, like deformable models.

In the deformable models approach curves in 2D images (or surfaces in 3D ones) are seen as objects able to deform under the action of particular forces. More precisely the curve or surface obtained from the segmentation process is the result of an optimization process aiming at the minimization of an energy functional associated to the problem. This energy is generally given by the sum of an internal energy and an external energy. The first component is entitled to preserve the regularity of the shape, while the second one maintains the adherence with the image features, governing the local velocity of deformation.

If we focus on a 2D example, a contour can be represented as $C(s) = [x(s), y(s)]^T$, where $s \in [0, 1]$ is a parameter and $x, y$ are the coordinate functions. The general form of the internal energy is

$$\varepsilon_{int}(C) = \int_0^1 w_1(s) \left| \frac{\partial C}{\partial s} \right|^2 + w_2(s) \left| \frac{\partial^2 C}{\partial s^2} \right|^2 ds$$

when the contour is modeled as a stretchy and flexible medium. The functions $w_1(s)$ and $w_2(s)$ control the tension and the stiffness of the contour respectively. The external energy is expressed as

$$\varepsilon_{ext}(C) = \int_0^1 P(C(s)) ds,$$

where $P(x, y)$ is a scalar potential depending on the image and can be chosen as proportional to the negative magnitude of the gradient of the image intensity $I(x, y)$ [53, 93].

Deformable models are defined to be explicit if the desired curve or surface is described explicitly, through a proper parametric representation, while they are named implicit if the sought object is an isoline or isosurface (level set) of another function (embedding function) which is explicitly described.

The energy minimization problem can also be reformulated as a dynamical system, governed by the energy functional, and evolving to its equilibrium, representing a local minimum of the energy. In this way also the evolution in time of the shape can be evaluated.
For 2D explicit deformable models, the typical evolution equation has the following expression:

\[
\frac{\partial C}{\partial t} = w_1 \frac{\partial^2 C}{\partial \chi^2} - w_2 \frac{\partial^4 C}{\partial \chi^4} - w_3 \nabla P(C)
\]  

(4.1)

the unknown of the problem being the contour itself and \( w_3 \) being a proper weight.

The evolution equation for an implicit model is based on the relation between the embedding function \( F : \mathbb{R}^3 \rightarrow \mathbb{R} \) and its level set \( C \). By definition, \( C \) remains a level set of \( F \) over time. Consequently,

\[
\frac{\partial F(C)}{\partial t} = -\nabla F(C) \cdot \frac{\partial C}{\partial t} = -|\nabla F(C)| \frac{\partial C}{\partial t} \cdot \frac{\nabla F}{|\nabla F|}.
\]  

(4.2)

By plugging the expression of the time derivative of \( C \) given in (4.1) into (4.2), the evolution equation for implicit deformable models is obtained.

The main advantages of the explicit approaches are the straightforward description of the interested object, that can be also used to track the motion of each point during the process, and the limited computational effort required. The main disadvantage is that if the deformation of the initialized object into the target involves a change in the topology or large displacements, a re-parametrization of the curve/surface may be necessary. On the other hand the main advantage of implicit approaches is that they are able to manage at no additional costs any changes in the topology of the segmented object or large deformations, since they don’t affect the embedding function. This feature is particularly important in vessels segmentation, due to the presence of several branches in the geometry. However, this is of course obtained with a remarkably higher computational cost with respect to explicit models.

Examples of explicit deformable models in 2D are discrete dynamic contours and deformable splines, usually called snakes, while in three dimensions the same approach is used in the balloons method. Since implicit deformable models aim at finding an isoline or isosurface of the embedding function, the corresponding techniques are referred to as level set methods. For a comprehensive review on deformable models we refer to [93] while for a detailed description of the level set method in particular we refer to [144].

The image segmentation process used for building the geometry models in this work has been carried out using Vascular Modeling ToolKit (VMTK). VMTK has been developed for performing 3D reconstruction, geometric analysis, mesh generation and surface data analysis for image-based modeling of blood vessels, but can be effectively used also to segment cardiac geometries.

In particular, segmentation from vessels CT and from cardiac MRI images (more precisely the extraction of the endocardium surface) used in the present thesis has been performed using the VMTK implementation of the level set method. The reconstruction of the left ventricle wall from SPECT images (see Chapter 3) features a very sharp...
contrast, has been done by extracting a proper level set of the image intensity function, through the marching cubes method [89].

In all these cases the output of the reconstruction process is a triangulated surface expressed in stl format, which provides the normal vectors and vertices of each triangular facet in a 3D Cartesian coordinate system.

The tetrahedral volume meshes needed for the 3D numerical simulations have been then generated using either Cubit\(^3\), Netgen\(^4\) or Tetgen\(^5\) software.

### 4.2 Registration algorithms

Image registration is a process that aims at determining a spatial transformation that relates position of landmarks in one image, to corresponding positions in one or more other images.

This technique may have various applications in biomedical sciences: it can be used for instance for merging the information obtained from different viewpoints or from different imaging devices, in order to retrieve more complete information on the patient. It can also be used to map an anatomy atlas on individual subjects imaging data, with the purpose of detecting potential abnormalities. The application that is considered in this work is the alignment of images captured at different time instants in order to track the motion of each point of the geometry.

In this section we introduce the main features of a general registration algorithm and briefly review a possible classification reported in [92]. For further references on image registration we suggest [37], [49] and [6].

The main components of a generic registration algorithm are a proper similarity measure, used to quantify the matching of two images, a transformation model, which specifies how the source image can be modified to match the target depending on a set of parameters, and the optimization process that varies the transformation model parameters to satisfy the matching criterion.

The existing registration methods can largely differ, depending on the different models on which they are based or on the different algorithms used to solve the resulting mathematical problem. Different models and methods are appropriate for different applications. We report here some criteria that can be used to classify a registration method, and to summarize the main properties of the algorithms available in the literature.

1. **spatial dimensionality** of the involved medical images (source-target), yielding 2D-2D, 3D-3D or 2D-3D registration algorithms;

2. **starting point of the registration**, which can be:
   a) images obtained by placing on the patient (in invasive or non-invasive way) proper artificial objects used as markers in the registration;

---

\(^3\)http://cubit.sandia.gov/
\(^4\)http://www.hpfem.jku.at/netgen/
\(^5\)http://tetgen.berlios.de/
4 Simulation of biophysical phenomena in moving domains based on medical images

b) images containing geometrical or anatomical landmarks;
c) segmented curves or surfaces;
d) intensity functions defined on raw images pixels/voxels;
e) coordinate systems of the acquisition devices used;

3. nature of the transformation, which can be for instance rigid or affine;

4. extent of the interaction with the user;

5. optimization procedure adopted;

6. subjects from which the images are acquired, which can be a single patient or different patients or a patient and a model;

7. anatomical region covered by the acquisition and registration;

8. imaging modalities of the source and the target of the registration. They can be acquired with the same modality, producing monomodal registration, or with different modalities, yielding multimodal registration. An image can also be registered to a template model.

The set of registered vessel surfaces employed in Section 4.3 for performing fluid dynamics simulations in a moving domain has been produced by applying a multilevel registration method. We provide here just a qualitative idea on the method, since it has not been developed in the context of this work. We refer to [97] and [121] for further details on this topic. This algorithm takes as input two segmented surfaces, a source surface and a target one, that in our case correspond to images acquired at subsequent time frames, and finds the transformed source cloud of points that best approximates the target cloud of points. The surface registration is formulated as a large-scale, non-convex optimization problem. The functional to be minimized is composed by the sum of a measure of the distance between the two clouds of points and of a regularizing term. The latter is added to obtain a realistic deformation model, penalizing undesirable deformations while allowing for “reasonable” ones. Different regularizations can be used (see [97]). For this application a regularizing term coming from a physical model of the wall is considered. In particular, the energy of an elastic thin membrane is a suitable regularization for the vessel. However, a good trade-off between accuracy and computational cost motivates the use of a mass-spring model as a viable alternative. The optimization problem is solved by using a multilevel algorithm. This algorithm achieves good performances on vessel geometries, while it is less suitable for the registration of tissues that undergo large deformations.

For this reason during this thesis a different registration algorithm has been implemented and applied to 2D MRI images of the heart. In Section 4.2.1 we describe the

---

6This registration algorithm has been developed and implemented by Dr. E. Haber (Emory University, Atlanta, GA, USA, and University of British Columbia, Vancouver, BC, Canada), Dr. S. Heldmann (University of Lübeck, Germany) and M. Piccinelli (Emory University, Atlanta, GA, USA).
formulation of this algorithm and its properties, we show some numerical results and we propose an approach for applying this method to track the motion of a segmented contour.

4.2.1 Non-rigid viscous fluid registration algorithm

The goal of this approach is to find a map that aligns a 2D or 3D template image with a topologically similar target image by applying a vector field transformation to the material coordinate system of the template and constraining the transformation to be smooth even when dealing with large deformations. It has been proposed in [27] and [26] for registering images of neuroanatomical structures with a template model available.

Mathematical model

This registration algorithm works directly on the intensity functions recorded on the images, instead of working on segmented surfaces or contours. In particular it needs a template image intensity \( T(x) \), with \( x \in \Omega \) and a target image intensity \( S(x) \). The spatial domain \( \Omega \) can be either contained in \( \mathbb{R}^2 \) or in \( \mathbb{R}^3 \), depending on the available images. The algorithm looks for a displacement \( d(x,t) : \Omega \times [0, \tau_{\text{conv}}] \rightarrow \Omega \) that minimizes a suitable distance measure between the deformed template \( T(x - d(x)) \) and the target image \( S(x) \), subject to constraints induced by a proper mechanical model. Here \( \tau_{\text{conv}} \) is a parameter whose meaning will be clarified in the sequel. The mechanical model is not used in the deformation process to describe the physical motion of the tissue. A mechanical analogy is used instead to enforce topological properties suitable for large deformation non-linear kinematics. To accomplish this property, the template is modeled as a highly viscous fluid, instead of using the more common assumption of linear elasticity or thin plates mechanical models.

The problem is set in an Eulerian reference frame, based on the pixel grid of the image, and the displacement field \( d(x) \) is defined as a map from points in the template to fixed observation points in the deforming continuum.

The relationship between the material velocity, \( u \), and the displacement is

\[
\frac{d}{dt} = \frac{\partial d}{\partial t} + (u \cdot \nabla) d,
\]

where \( \frac{d}{dt} \) denotes the total derivative.

The model for the continuum deformation is then obtained by imposing the conservation of the linear momentum, expressed in (2.25), which gives the relationship between the applied body forces (renamed \( b \) in this context), the state of stress and the resulting material deformation. The material is modeled to be a Navier-Poisson Newtonian fluid and its constitutive law is expressed in equation (2.30). The momentum conservation for this kind of fluid is expressed in equation (2.31). The registration algorithm at hand solves a simplified model valid for very low Reynolds numbers, obtained by neglecting the pressure gradient and the inertial terms. The PDE to solve is therefore

\[
-\mu \Delta u - (\lambda + \mu) \nabla (\nabla \cdot u) = b(d)
\]
4 Simulation of biophysical phenomena in moving domains based on medical images

with homogeneous boundary conditions on the velocity, assuming the template and target images to be already aligned on their boundaries.

The first term in (4.4) accounts for constant volume viscous flow of the template, while the second one accounts for local changes of volume. The body force links the target image and the deforming continuum, driving the deformation of the template to match the target. It is the gradient of a cost function that minimizes the squared difference between the intensities of the two images. The resulting expression for $b$ is

$$b(x, d(x, t)) = \alpha \left( T(x - d(x, t)) - S(x) \right) \nabla T\big|_{x-d(x, t)}.$$  

The term $\nabla T\big|_{x-d(x, t)}$ is the gradient of the template evaluated on the new configuration, it presents larger values at the edges of the structures in the image, and it determines the direction of the deformation forces. This term is weighted by the difference between the deformed image and the target one, producing a higher magnitude force in regions with large mismatch. When the template is perfectly aligned with the target, the body force will be coherently zero.

The problem to solve is therefore

$$\begin{align*}
-\mu \Delta u - (\lambda + \mu) \nabla (\nabla \cdot u) &= \alpha \left( T(x - d(x, t)) - S(x) \right) \nabla T\big|_{x-d(x, t)} \quad x \in \Omega, t \in [0, \tau_{\text{conv}}] \\
\frac{\partial d}{\partial t} + (u \cdot \nabla) d &= u \quad x \in \Omega, t \in [0, \tau_{\text{conv}}]
\end{align*}$$  

(4.5)

A possible justification of this approach comes from interpreting the solution of the registration process as the transformation that maximizes the posterior distribution (MAP) of the image, in a Bayesian framework. The basic idea of this approach, as explained in [79], is to consider the deforming template image as an array of pixels $f$ associated with a prior probability $p(f)$ and the target image as an array of observed data $g$ on the same pixels. In Bayesian statistics, the probability $p(f|g)$ of any image $f$ given the data $g$ is proportional to the product $p(f)p(g|f)$, where $p(g|f)$ is the probability of the observed data given $f$, called likelihood. The distribution of $p(f|g)$ is called posterior and is assumed to be in the form of Gibbs probability measure [70]. The total potential associated to the posterior is then given by the sum of the likelihood contribution and the prior contribution. Since in [26] the likelihood is modeled as a Gaussian distribution, its contribution to the Gibbs potential is proportional to $\int_\Omega (T(x - d(x, t)) - S(x))^2 \, dx$ and represents the driving force triggering the deformation process. According to this interpretation, the prior contribution includes the desired properties of the deformation and can be chosen as a suitable mechanical potential.

In this framework the PDE formulation of the problem stems from minimizing the Gibbs potential. The momentum conservation equation corresponds to the variation of the prior distribution with respect to the displacement field and the forcing term is the variation of the likelihood function. The theoretical basis of this approach is described in [27] and founded on variational principles explained in [96].

The peculiarity of the fluid registration approach is the choice of a viscous fluid prior distribution, instead of the more classical linear elastic solid or thin plate prior distribution. Its advantage is that the regularized field in the transformation is the velocity,
instead of the displacement, since the stress tensor depends on the velocity. Therefore, when the stress restraining the motion relaxes over time, large-magnitude deformations are allowed, while in standard elastic registration, large deformation are penalized by a restoring force that increases with displacement.

Another advantage of this approach is that it is image-based and therefore does not require an image segmentation for each time frame. This aspect is particularly important in cases in which the segmentation process needs interactions with an operator.

The main disadvantage is the high computational cost, since the PDEs system has to be solved in all the volume defined by the bounding box of the surface for 3D registration.

In our application, defining $\Delta t$ as the time frame between two subsequent acquisitions and $T_{end}$ the time of the last acquisition, the template image will be the image captured at time $t = (k+1) \Delta t$ and the target image will be the one acquired at $t = k \Delta t$, and we will perform a registration procedure for $k = 0, 1, T_{end} - 1$.

**Numerical algorithm**

While in [27] a finite difference scheme is employed for the solution of the equations (4.5), here the problem has been reformulated in a finite element framework.

The variational formulation of (4.5) reads: given $T(x)$ and $S(x) \in V$, find $u \in L^2(0, \tau_{conv}; V)$ and $d \in H^1(0, \tau_{conv}; V)$, with $V = [H^1(\Omega)]^3$, such that

\[
\begin{aligned}
&\int_{\Omega} \mu \nabla u \cdot \nabla \varphi - \int_{\partial \Omega} \mu (\nabla u \cdot n) \cdot \varphi + \int_{\Omega} (\lambda + \mu) (\nabla \cdot u) (\nabla \cdot \varphi) + \\
&- \int_{\partial \Omega} (\lambda + \mu) (\nabla \cdot u) (n \cdot \varphi) = \int_{\Omega} b(d) \cdot \varphi \\
&\int_{\Omega} \frac{\partial d}{\partial \tau} \cdot \psi + \int_{\Omega} ((u \cdot \nabla) d) \cdot \psi - \int_{\Omega} u \cdot \psi = 0 \\
u(x, 0) = u_0 \\
u(x, \tau)|_{\partial \Omega} = 0,
\end{aligned}
\]

for each $\varphi$ and $\psi \in V$. Due to the boundary condition on $u$, the boundary integrals on $\partial \Omega$ vanish.

We remark that the time variable of the original mechanical model does not have a physical meaning in this context and it is not related to the acquisition time. It can instead be interpreted as a counter of the iterations necessary to achieve the alignment between the deforming template and the target. Therefore here we denote it with a different symbol $\tau$. $\tau_{conv}$ is the minimum value of $\tau$ in which the chosen stopping criterion is fulfilled.

In the employed advancing scheme, for each $\tau^{n+1} = \delta \tau (n + 1)$ we solve

\[
\int_{\Omega} \mu \nabla u^{n+1} \cdot \nabla \varphi + \int_{\Omega} (\lambda + \mu) (\nabla \cdot u^{n+1}) (\nabla \cdot \varphi) = \int_{\Omega} b(d^n) \cdot \varphi \quad x \in \Omega,
\]

(4.7)
looking for $u^{n+1} = u(\tau^{n+1})$. We then compute $d^{n+1} = d(\tau^{n+1})$, using $u^{n+1}$ to solve
\[ \int_{\Omega} \frac{d^{n+1} - d^n}{\delta \tau} \cdot \psi + \int_{\Omega} \left( (u^{n+1} \cdot \nabla) d^* \right) \cdot \psi = \int_{\Omega} u^{n+1} \cdot \psi. \] (4.8)

The symbol $*$ stands for either $n$ or $n+1$ according to the choice of an explicit or implicit scheme for (4.8).

Both equations (4.7) and (4.8) are discretized in space in a finite element approximation framework, choosing the $P_1$ finite element space as an approximation of $V$.

As a stopping criterion of this iterative scheme we check if the $l^2$ norm of
\[ T(x - d(x, \tau)) - S(x), \]
normalized over the $l^2$ norm of $S$, is below a certain tolerance and if this quantity increases after reaching a local minimum. If at least one of these two stopping criteria is satisfied, we stop the iterations.

The solver for this problem has been implemented in LifeV software [1] and details about the implementation are given in Section 5.4. We mention here that the linear system obtained after “time” and space discretization of (4.7) is solved with the Conjugate Gradient method, while (4.8) is solved with CG method when $* = n$ and with GMRES method if $* = n + 1$, the corresponding coefficient matrix being non-symmetric with the implicit treatment of the convective term.

Numerical results

We applied this algorithm to the registration of a time series of 2D MRI images of the heart, acquired with a Philips Medical Systems SSFP MRI machine. From a data set composed of 11 short axis slices, acquired at 20 different time frames during the cardiac cycle, we chose a slice corresponding to the upper part of the ventricles and performed 20 registration procedures between each pair of subsequent images in time. In this case two consecutive acquisitions represent the biological structure in two time frames, separated by an interval $\Delta t = 40$ ms. Since we consider one cardiac beat, we assume a periodic dynamic so that the 20th registration procedure is performed between the 20th image and the first one. In the original MRI data set an entire torso was represented. To save computational time the images have been pre-processed using Matlab software, to trim the displayed region around the area of interest.

We present now the preliminary results obtained with this technique. The first numerical tests we report have been performed to tune the parameters of the model. In Figure 4.1 we show the dependence on the iteration count of
\[ \chi = \frac{\|T(x - d(x, \tau)) - S(x)\|_2}{\|S(x)\|_2} \]
representing a normalized dissimilarity index.

---

7These images are courtesy of Dr. John Oshinski (Emory University and GeorgiaTech, Atlanta, GA USA).
8http://www.mathworks.com/
This test shows that, if a bigger $\delta \tau$ parameter is used, the explicit algorithm ends prematurely (dash-dotted line with stars), without achieving the desired solution. With the smallest $\delta \tau$, on the other hand, both the implicit and explicit schemes (solid line and wide dotted line, respectively) converge to the fixed threshold (which in this case is 0.07) and this happens in both cases with less iterations than in the case of the implicit scheme with a bigger $\delta \tau$.

We also tried different values of $\mu$ with both implicit and explicit schemes. The result, presented in Figure 4.2, shows that keeping the same ratio between $\mu$ and $\delta \tau$ produced the same rate of convergence and that for the chosen $\delta \tau$ both the implicit and the explicit schemes converge. In this situation the explicit scheme is preferred due to its smaller demand in computational time.

Fixing $\mu = 2000$ a test on the dependence of the convergence rate on the parameter $\lambda$ has been carried out. We have performed simulations with values of lambda in the range $[-2000, 2000]$, including the incompressible case with $\lambda = -2000$. The results in Figure 4.3 show that the incompressible cases (thick dashed and thick dotted lines) do not converge to the desired threshold and that the fastest convergence is achieved with $\lambda = -500$ (solid line with stars).

We show in Figure 4.4 the results of the registration of the selected slice applied to different time frames in the cardiac cycle. The left column shows the template image corresponding to $t = (k+1)\Delta t$, the middle column shows the target image corresponding to $t = k\Delta t$ and the right column shows the result of the transformation, that is $T(x - d(x, \tau_{\text{conv}}))$. The white circles highlight the area where the deformation is more evident.
4 Simulation of biophysical phenomena in moving domains based on medical images

Finally we performed a simple segmentation process\(^9\) on the slice corresponding to the first acquisition time, to extract the contour of the left ventricle endocardium. We then transformed each point of the segmented curve according to the displacement field obtained with the registration process. In particular we applied to each point of the curve the transformation

\[
h(x, t^k) = x - d^k(x, \tau_{\text{conv}}),
\]

\(d^k(x, \tau_{\text{conv}})\) being the displacement computed with the registration between \(t^k\) and \(t^{k+1}\), and we iterated the procedure with \(k\) going from \(k = 1\) to \(k = 20\). Since in general the points of the segmented curve may not coincide with nodes in the fluid mesh, a linear interpolation is performed of the displacement field. As a result, the contour contracts following the real, image-based dynamics of the myocardium during a heart beat. In Figure 4.5 we show some screenshots of this reconstructed movement.

**Future developments**

Thanks to its particular formulation, this registration approach seems promising to reconstruct the motion of the cardiac tissue. So far, only a 2D example has been analyzed,

\(^9\)At this preliminary stage, we performed the segmentation of a contour curve using Matlab (http://www.mathworks.com/).

104
Figure 4.3: Dependence of $\chi = \frac{\|T(x-d(x,\tau))-S(x)\|_2}{\|S(x)\|_2}$ on the iteration count for different schemes and values of $\lambda$. In the legend “impl” stands for implicit and “expl” stands for explicit.

since the 3D version of the finite element solver for this problem is currently under testing process.

A future development of this work will include the validation of the approach on “artificial” images generated by applying a known displacement to a given image and then using the fluid registration algorithm to retrieve the displacement.

After that the reliability of the method will be proved, it will be applied on a complete set of short-axis images of the heart, or directly to the whole volume of acquisition. Once the displacement field is defined for each point of the volume mesh and for each acquisition time frame, the same procedure accomplished on the segmented endocardium contour will be applied to the segmented endocardium and epicardium surfaces, obtaining the tracking in time of the heart surfaces motion during a heart beat.

In the next Section we will describe how the information on the motion of the geometry (vessel or heart) surfaces can be employed to perform numerical simulations in a moving domain.

4.2.2 Moving domain

Let us now suppose that a 3D closed surface has been segmented and the movement of each point of the surface has been tracked in time, with a proper registration algo-
Figure 4.4: Results of the fluid registration algorithm applied to a 2D heart short-axis MRI at different acquisition time frames. The left column shows the template image corresponding to $t = (k + 1)\Delta t$, i.e. $T(x)$, the middle column shows the target image corresponding to $t = k\Delta t$, i.e. $S(x)$, and the right column shows the result of the transformation $T(x - d(x, \tau_{\text{conv}}))$. On the first row $k = 1$, on the second $k = 5$ and on the third $k = 19$. The white circles, where present, highlight the area where the deformation is more evident. The two large regions in red represent the ventricular cavities, surrounded by the ventricular wall (light blue).

\footnote{The same discussion applies to the case of a 2D closed curve, but for the sake of clarity we will focus hereby on the 3D case.}
In general, due to technical limitations on the imaging devices, the number of possible acquisitions during a cardiac cycle has the order of magnitude of 10, while numerical simulations of both the action potential propagation and the fluid dynamics in vessels require a much finer time discretization, to provide reliable results.

The available displacement field needs therefore to be interpolated in time, to obtain a displacement vector for each simulation time step and each surface node. The simplest choice is a piecewise linear interpolation, but it does not guarantee the continuity of the...
time derivative of the interpolated field, and it can consequently produce artifacts in the simulated variables. To avoid this problem we propose to compute a spline interpolation of the available data.

Once the surface displacement field is available, it is necessary to track in time the movement of each point of the domain to perform a numerical simulation in the enclosed moving domain. The most common approach in the literature is to compute, for each time step of the simulation, the harmonic extension in the 3D domain of the displacement (or velocity) values computed on its enclosing surface by the registration step. The goal of this phase is to extend the known surface movement to a regular field defined in the interior of the domain (see e.g. [42,88,109]).

In this approach we solve at each time step the following problem

\[
\begin{cases}
- \Delta w = 0 & x \in \Omega(t), t \in (0,T] \\
w(x, t)|_{\partial \Omega} = w_{\Gamma}(t) & t \in (0,T],
\end{cases}
\tag{4.9}
\]

where \(w\) represents the domain velocity and its boundary conditions \(w_{\Gamma}(t)\) is the time derivative of the surface displacement field obtained with the registration step.

In our test cases, problem (4.9) is solved employing a finite element piecewise linear approximation, and the linear system solution is computed using the Conjugate Gradient method. The obtained mesh velocity field is then ready to be included in the moving domain formulation of the considered problem (see problems (2.32) and (2.22)).

4.3 Application to real cases

4.3.1 Moving aorta

The approach introduced in this chapter has been applied to simulate blood flow in a moving aorta\(^{11}\). The dataset of medical images has been acquired at Ospedale Maggiore in Milano, using a Siemens SOMATOM Definition Flash Dual-Source CT Scanner, able to capture 10 time frames per cardiac cycle. From the original scanned volume, the region including the aortic arch and the thoracic aorta has been selected and a level set segmentation for each time frame has been performed, as explained in Section 4.1.2. All the side branches have been excluded from the domain of interest, depicted in Figure 4.6.

The motion of the segmented surface has been tracked in time using the multilevel registration algorithm briefly described in Section 4.2. Using the software Cubit\(^{12}\) we generated a tetrahedral mesh in the domain enclosed by the surface geometry corresponding to the first image in the data set. The movement of the mesh points was computed as the harmonic extension of the wall movement, obtained by the registration.

\(^{11}\)This part of the work, as the one described in Section 4.4, has been carried out in collaboration with Dr. Tiziano Passerini (Emory University, Atlanta, GA, USA).

\(^{12}\)http://cubit.sandia.gov/
The finite element formulation of problem (2.32) has been solved with an ad hoc software based on LifeV library [1], choosing to approximate both the fluid velocity and the pressure with piecewise linear functions. More precisely, the solution of the Navier-Stokes equations in ALE formulation was achieved with a solver based on an Interior Penalty (IP) stabilization technique [21], that does not constrain the choice of LBB compatible elements for velocity and pressure.

Since in this case patient-specific measurements of blood flow rates, velocity or pressures were not available, we imposed an analytical boundary condition for the velocity on the inlet section. In particular the velocity profile on the section is parabolic and its amplitude in time has been tuned in order to get a half-sinusoidal flux in systole and a small constant flux in diastole, in a physiological range (see [38]). Moreover, we measured the time pattern of the area of the inlet section from the segmented geometries, and we synchronized the flux peak with the measured inlet area peak. On the outlet section we imposed a homogeneous Neumann boundary condition. The inflow flux in diastole was not set exactly to zero to avoid remarkable back-flow due to the change in volume of the moving vessel. This problem was observed with the choice of an analytical inflow boundary condition but would most likely be avoided if measured patient-specific flow conditions were available, since we expect them to be completely compatible with the actual volume dynamics.

The boundary conditions here prescribed are not intended to reproduce the actual blood flow in the considered patient. Nevertheless they are used to evaluate the effects of the wall motion on the results of numerical simulations of fluid dynamics and to test a pipeline for performing image-based motion CFD. In case measured boundary conditions
were available, they could be immediately taken into account in the simulation.

We simulated 3 cardiac cycles after a preload phase and we show here some results extracted from the last one. The duration of each simulated cardiac cycle is 0.82 s, the time step of the simulation is $8.2 \times 10^{-4}$ s and the computational mesh is composed of 9160 vertices. The fluid viscosity is $\mu = 0.03$ poise and the fluid density is 1 g/cm$^3$.

In Figure 4.7 we show the domain velocity on the mesh boundary at $t$ corresponding to the initial time step of the cycle, to $\frac{1}{4}$, $\frac{1}{2}$ and $\frac{3}{4}$ of the cardiac cycle. As expected, due to its proximity to the heart, the first tract of the aortic arch is interested by the most noticeable movement, while the distal part of the domain presents relatively small movements.

To highlight the effects of the moving walls on the computed fluid dynamics, we also performed a simulation in the geometry of the aorta under the assumption of rigid vessel walls, the position of the mesh nodes being kept fixed in the configuration corresponding to the initial time step of the cycle. All the other parameters, together with inlet and outlet boundary conditions, are chosen as in the moving wall simulation. The assumption of fixed vessel walls is indeed typically exploited in CFD modeling of blood flow problems, due to a lack of information regarding the response, the thickness of the vessel and the intra-arterial pressure waveform [39].

To evaluate the difference between the two solutions, we mapped the points of the moving geometry back to the initial configuration, obtaining a comparison on a reference domain.

We show in Figure 4.8 the difference between the fluid velocity field of the fixed domain and the moving domain simulation, on selected sections of the aorta at various time steps. The Figure is divided in six panels, each panel being formed of a couple of images separated by a black vertical line. On the left of each panel we show the velocity difference field, while on the right we report the velocity field in the moving domain. We notice that the magnitude of the difference is larger during the early systole and the late diastole than in the central part of the cycle. This behavior can be explained considering that when the inflow blood flow is higher (i.e. around the peak systole), the fluid dynamics in the aorta is governed by the boundary condition prescribed, while when the inflow velocity flux is lower, the fluid dynamics is mostly influenced by the wall motion.

The first tract of the aortic arch and the downstream region present the most remarkable differences in the velocity field. In the former this is likely due to the significant displacement that the region is experiencing. In the latter it can be due to the presence of a small back-flow in the moving domain simulation, which is absent in the fixed one and has to be considered, as already mentioned, an artifact related to the lack of patient specific boundary conditions.

We also compared the moving domain and fixed domain simulations in terms of the wall shear stress (WSS), defined as the tangential component of the stress exerted by the fluid over the vessel wall $\Gamma_{\text{wall}} \subset \partial \Omega$:

$$WSS = (\sigma n)_t = \sigma n - (\sigma n \cdot n)n$$

In the previous expression $n$ is the normal unit vector on $\Gamma_{\text{wall}}$ and $\sigma$ is the Cauchy stress.
Figure 4.7: Screenshots displaying the velocity of the domain at different time in the cardiac cycle, in different panels, each of them identified by a black vertical line in the middle. Let $T$ be the cardiac time period. The first row shows the results at $t = 0$ (left panel) and at $t = \frac{T}{4}$ (right panel); the second row shows the results for $t = \frac{T}{2}$ (left panel) and $\frac{3T}{4}$ (right panel). The magnitude of the vector field is represented by the color scale, while its direction is indicated by the arrows. The considered measurement units for the velocity is cm/s.

tensor for an incompressible Newtonian fluid, defined in (2.26). In our simulations the WSS is computed with a software based on LifeV\textsuperscript{13} by performing a post process procedure. More precisely, the wall shear stress field over the computational domain boundaries is computed exploiting a $L^2$ approximation of the velocity gradient, given a precomputed velocity field (see [114] and Section 5.5).

The results concerning the comparison of WSS in fixed and moving domains are

\textsuperscript{13}This part of the code has been developed by Dr. T. Passerini (Emory University).
shown in Figure 4.9 for two different perspectives and at \( t \) corresponding to peak systole. A remarkable difference in WSS appears on the inner side of the aortic arch curve. Significant differences are also found in the ascending tract of the aorta and in the downstream region. This is likely due to the large displacement occurring in this regions at peak systole, clearly shown in Figure 4.7.

To quantify the differences between the solution fields in the fixed domain simulation (denoted by subscript \( \text{fixed} \)) and the moving domain solution fields (denoted by subscript \( \text{moving} \)), we computed the indicators

\[
\chi_{f,L^2} = \frac{\|f_{\text{fixed}} - f_{\text{moving}}\|_{L^\infty(0,T;L^2(\Omega))}}{\|f_{\text{fixed}}\|_{L^\infty(0,T;L^2(\Omega))}}
\]

\[
\chi_{f,H^1} = \frac{\|f_{\text{fixed}} - f_{\text{moving}}\|_{L^\infty(0,T;H^1(\Omega))}}{\|f_{\text{fixed}}\|_{L^\infty(0,T;H^1(\Omega))}}
\]

(4.10)

where \( f \) is in turn the fluid velocity or the fluid WSS.

The results of this analysis show a considerable difference between the moving fields and the fixed fields. More precisely we get \( \chi_{u_L^2} = 0.1115 \) and \( \chi_{u,H^1} = 0.1381 \) for the velocity and \( \chi_{WSS,L^2} = 0.2250 \) and \( \chi_{WSS,H^1} = 0.2706 \) for the wall shear stress.

These results suggest that in the considered vascular district the wall motion has a noticeable effect on the solution, the differences being more than 20% for the WSS and more than 10% for the velocity. A different conclusion on a similar numerical experiment was drawn in [39], where blood flow in intracranial aneurysms was considered and simulations in moving domains were found to provide results qualitatively in good agreement with those obtained in rigid domain simulations. In that case, however, the limited motion of the considered vascular region was likely responsible for the different observed outcome. In facts, it follows from the observation of our preliminary results that the arterial wall motion may not be neglected in some vascular districts, and in those cases the image-based technique could provide a relatively easy way to improve the accuracy of CFD simulations.

### 4.3.2 Moving left ventricle

This Section concerns the application of the image-based strategy to numerical simulations of the action potential propagation in a moving left ventricle wall.

The 4D dataset of available medical images was described at the beginning of Section 4.2.1. The endocardium surface has been segmented using the level set method, as explained in Section 4.1.2, while the external surface of the domain of interest, including part of the epicardium surface and the sectum, has been manually segmented. The domain of interest is depicted in Figure 4.10.

The motion of the segmented surfaces has been tracked in time using the multilevel registration algorithm presented in Section 4.2. Then the segmented surfaces corresponding to \( t = 0 \) s have been capped with a horizontal plane, to obtain a closed domain in which a tetrahedral mesh has been built, using Cubit\(^{14} \) software. During the simulation, the configuration of the mesh was updated by means of the harmonic extension of the wall movement, computed by the registration.

\(^{14}\text{http://cubit.sandia.gov/}\)
No estimation of the position and direction of the cardiac fibers was available for the considered geometry. Therefore the analytical description of the fiber direction mentioned in Section 3.2.4 has been employed, devising a proper mapping of points in the ellipsoidal geometrical to points in the actual MRI based geometry.

To simulate the action potential propagation we here couple the Bidomain tissue model with the Rogers-McCulloch ionic model, adopting a \( P_1 \) finite element approximation. The time advancing scheme has already been described in Chapter 3 and the linear solver chosen for this test is the CG iterative method preconditioned with an ILU preconditioner. The problem has been set in a moving framework by updating the position of points in the computational mesh during the simulation and deforming coherently the fibers vectors, on the basis of the computed deformation gradient, as explained in Section 2.1.2. As a consequence, the conductivity tensors and the stiffness matrices need to be updated every time that a non negligible deformation occurs.

The time step of the simulation is 0.1 ms and the other parameters involved in the models have the same values as in Chapter 3. The computational mesh is composed if 9160 vertices.

Some screenshots of the simulations are collected in Figure 4.11. Even though the simulation ran for 400 time steps, showing a reasonable pattern for the action potential in the moving domain, after \( t = 43 \) ms the quality of the mesh degenerated abruptly close to the edge between the artificial cap and the endocardium surface (Figure 4.12), producing a blow up in the local transmembrane potential, which caused the simulation to end prematurely.

A possible explanation for this failure is that topological constraints were not forced on the tetrahedra of the computational domain, since the registration process was performed only on the boundary surfaces. This choice limited the range of deformations that could be smoothly tracked by this method, which was not designed to track large deformations.

A possible solution to this problem is to resort to the 3D version of the multilevel-registration algorithm, to allow for the tracking of volumes instead of surfaces. This would guarantee a good quality of the mesh. Preliminary experiments of this possibility showed however a significant increase in the computational cost, calling for an improvement of the performances of the algorithm. An extension of the present work would address this aspect. Another possibility is to resort to different registration algorithms, exhibiting good performances on myocardium motion tracking (e.g. [12]). For this purpose the fluid registration algorithm described in Section 4.2.1 will be further analyzed and tested.

### 4.4 Validation of the image-based simulations

Results presented in Section 4.3 suggest that performing numerical simulations of blood flow assuming rigid vessels walls, when the real wall movement is not negligible, can yield to noticeable differences in fluid velocity and wall shear stress fields with respect to the results of a moving domain simulation.
Nevertheless, to propose this approach as a possible alternative to more standard Fluid-Structure Interaction (FSI) modeling strategies we need to investigate its reliability. We therefore set up a comparison between results of the image-based moving domain and of a fluid-structure interaction approach.

The idea is to perform a FSI simulation on a given geometry, modeling the vessel wall with a suitable mechanical model, and the fluid mechanics as governed by the Navier-Stokes equations. From such simulation we can retrieve the fluid velocity and pressure fields that will be considered a benchmark and the displacement field of the computational domain. We then solve the Navier-Stokes equations in ALE formulation (2.32), using the results from the FSI simulation to provide boundary conditions for the velocity on the vessel wall, the mesh displacement and the ALE velocity $w$. We finally compare the values computed with the two strategies, namely the velocity and the wall shear stress fields. We report in Figure 4.13 a flow chart of the described pipeline.

The numerical experiments proposed here can be in principle realized with any wall mechanics models, fluid-structure algorithms and geometries. Its practical application is only limited by computational time and technical difficulties for developing the desired fluid-structure solver, depending on a model for the wall mechanics.

Many different fluid-structure approaches are present in the literature. See for example [9, 53, 109, 159]. In the LifeV software project, both segregated and monolithic approaches have been implemented to simulate fluid-structure interaction in vessels (see [48, 52]).

In particular, recently a monolithic, also defined non-modular, solver for a fluid-structure interaction model was added to the software library. The considered model is composed of St.Venant-Kirchhoff mechanic model (2.35) plugged in (2.34), and Navier-Stokes equations (2.32). The solver shows good properties of robustness with respect to the model parameters and good scalability in distributed computing environments, taking advantage of ad-hoc preconditioners [36], and was here chosen to validate the image-based approach. We refer to the cited paper for all the details about the formulation of the problem and the implementation of the solver. We report here only the interface conditions that are enforced to the two coupled models and the time discretization scheme adopted since the set up of a correct comparison is strongly dependent on these choices. In particular, at the interface between the fluid domain and the structure domain the algorithm imposes the continuity of the velocity and the continuity of the stress. It also requires the geometric adherence of the two domains. The fluid and structure problems are also coupled with a geometry problem that is the harmonic extension in the fluid domain of the solid displacement at the fluid-solid interface (see Section 4.2.2).

If we define $d_f$ the displacement of the fluid domain and $d_s$ the displacement field in the structure, we can express the fluid problem as

$$F(u, p, d_s, d_f) = 0,$$

(4.11)

This work has been performed by Paolo Crosetto (École Polytechnique Fédérale de Lausanne, Switzerland) and his collaborators.
where $F$ is an operator representing the Navier-Stokes equation (2.32) with interface conditions as specified above. The fluid problem is coupled with the geometry problem through the dependence of the ALE velocity $w$ on the fluid displacement $d_f$ and with the solid problem, through the interface condition on the continuity of the velocity.

In the same way we can express the solid problem as

$$S(u, p, d_s, d_f) = 0,$$  \hspace{1cm} (4.12)

where $S$ is the operator representing the structure equation (2.34). The interface condition prescribes the continuity of the stress, so that this problem depends on the fluid solution $(u, p)$ and on the fluid domain displacement.

Finally, we can introduce an operator $G$ representing the harmonic extension equation (4.9), and express the geometry problem as

$$G(d_s, d_f) = 0.$$  \hspace{1cm} (4.13)

With this notation a non-modular approach corresponds to solve the coupled non-linear system

$$\begin{align*}
F(u, p, d_s, d_f) &= 0 \\
S(u, p, d_s, d_f) &= 0 \\
G(d_s, d_f) &= 0.
\end{align*}$$  \hspace{1cm} (4.14)

Different time discretizations of the non-linear system can be employed, yielding solvers with different complexity, as described in [36]. We chose to use an explicit approach for the convective term in the fluid equation (2.27), which then reads $(u^n - w^n) \nabla u$. Moreover the fluid domain used is the one computed with the harmonic extension of the fluid displacement $d_f^n$ at the previous time step. In this way the fluid and solid problems can be decoupled from the geometry one, allowing to solve separately the two following subproblems

$$\begin{align*}
F(u, p, d_s, d_f^n) &= 0 \\
S(u, p, d_s, d_f^n) &= 0.
\end{align*}$$  \hspace{1cm} (4.15)

In the original set up of Section 4.3, the image-based fluid solver computes the boundary mesh velocity and moves the domain using the displacement at the current time step, since this data represents the most updated information available. To compare with a fluid-structure-geometry problem discretized in an explicit way, the image-based solver needs to be slightly modified. More precisely, in this part of the work, we move the mesh and compute the convective term according to the displacement at the previous time step, while the boundary condition on the velocity prescribed on the vessel wall is computed from the current interface displacement.

We perform a test on a geometry representing a carotid bifurcation. The fluid mesh has 20072 vertices and the solid one has 24100 vertices. On the inlet section we prescribe
for both the FSI and image-based movement simulation a non-homogeneous Neumann boundary condition, with the physical meaning of a normal stress and the time pattern of a step function. The stress magnitude is \(1.33 \times 10^4\) dyne/cm² for \(t < 0.003\) s and 0 otherwise. On the outlet sections we prescribe homogeneous Neumann boundary conditions. The time step for both simulations is \(\delta t = 10^{-4}\) s, the fluid viscosity is 0.03 poise, the fluid density is 1 g/cm³, the Young modulus for the structure problem is \(3 \times 10^6\) dyne/cm², the Poisson ratio is \(\nu = 0.3\) and the density of the structure is 1.2 g/cm³.

Both in the FSI and in the image-based motion approach, the finite element approximation of the fluid problem is adopted and all the unknowns are assumed to be functions in the \(P_1\) space. The fluid problem is stabilized as described in Section 4.3. We show in Figure 4.14 the results of the comparison of the fluid velocity fields at various time steps. The domain represented is a longitudinal section of the geometry. The figure is divided in five panels, each panel being formed of a couple of images separated by a black vertical line. Each panel displays on the left the magnitude of the difference field between the FSI velocity and the Image-Based (IB) motion velocity. On the right of each panel the IB motion velocity field is displayed. The analogous comparisons concerning the wall shear stress field is displayed in Figure 4.15. In this case the displayed domain is the lateral wall of the vessel.

The left side of each panel shows that the differences between the results obtained with the two strategy are small, especially with respect to the magnitude of the velocity field displayed in the right side of the panel. The same remark applies to Figure 4.15.

To obtain a precise evaluation of the error committed by the image-based approach compared to the FSI solution, chosen as the benchmark, we defined two indicators of the norm of the difference between the computed fields, analogous to those presented in (4.10):

\[
\zeta_{f,L^2} = \left\| \frac{f_{\text{FSI}} - f_{\text{IB}}}{\|f_{\text{FSI}}\|_{L^\infty(0,T;L^2(\Omega))}} \right\|_{L^\infty(0,T;L^2(\Omega))} \\
\zeta_{f,H^1} = \left\| \frac{f_{\text{FSI}} - f_{\text{IB}}}{\|f_{\text{FSI}}\|_{L^\infty(0,T;H^1(\Omega))}} \right\|_{L^\infty(0,T;H^1(\Omega))}
\]

\(f\) being either the velocity field or the WSS field.

The results of this comparison confirm that the differences between the FSI fields and the corresponding IB motion fields are small, the indicator being less than 0.03% for every variable and norm considered. In particular we get \(\zeta_{u,L^2} = 3.9678 \times 10^{-5}\) and \(\zeta_{u,H^1} = 2.1342 \times 10^{-4}\) for the velocity and \(\zeta_{WSS,L^2} = 4.0177 \times 10^{-5}\) and \(\zeta_{WSS,H^1} = 1.0012 \times 10^{-4}\) for the wall shear stress.

We performed the same kind of comparison also on a cylindrical geometry, obtaining comparable results.

Although we did not focus on tuning the two algorithms to improve their efficiency at this stage of the work, we observed a significant difference in the computational effort required to solve the two problems. This was indeed expected since the IB motion strategy does not involve the solution of the solid problem. The FSI code has been run using 8 processors while the image-based motion code has been run on one processor of the same machine\(^\text{16}\) since the IB motion algorithm is currently implemented only in

\(^{16}\)Workstation equipped with 2.2 GHz AMD Dual-Core Opteron processors and 4 GB RAM.
the serial version of LifeV. The CPU time required by each of the processors in the FSI simulation is approximately the double of the CPU time required by the image-based motion simulation. Besides this qualitative result, however, an extensive evaluation of the relative advantages of the two approaches is in order, exploiting in particular parallel computing on a large number of processors, to evaluate the scalability of the two algorithms to obtain a fairer comparison.

The preliminary results on errors and CPU time seem promising, since a considerable gain in computational resources can be obtained, losing only a negligible percentage in accuracy. Nevertheless, a more thorough evaluation of the image-based motion strategy will be object of our work in the next future. In particular we plan to investigate its performances on more complex geometries, involving more marked curvature and torsion and to study the effect of the space and time discretization parameters on the accuracy of the solution. The comparison with a different fluid-structure interaction algorithm, for instance involving a different mechanical model for the structure, can also be studied.

In this part of the work we focused on the application of the image-based motion technique to fluid-dynamics simulations. Provided that a set of registered volume meshes, with the required regularity, is available, the same technique can be used to simulate the action potential propagation in a moving heart. Again, the approach should be validated by comparing its results with those obtained by solving the electrocardiology problem coupled with a mechanical model for the myocardium. However, at this stage the LifeV library does not include a solver for the cardiac mechanics. We plan to implement such a feature and to realize the comparison as a development of this work.
Figure 4.8: Screenshots displaying, on the left side of each panel (identified by a black vertical line in the middle), the difference between the fluid velocity computed by the fixed domain simulation and the one computed by the moving domain simulation, on selected slices of the domain. On the right side the velocity computed by the moving domain simulation is depicted. The first row displays the screenshots at $t$ correspondent to the initial time step of the cycle (left) and to $\frac{1}{10}$ of the cardiac cycle (right); in the second row $t$ corresponds to $\frac{3}{10}$ (left) and $\frac{1}{2}$ (right) of the cardiac cycle; in the third row $t$ corresponds to $\frac{5}{6}$ (left) and $\frac{9}{10}$ (right) of the cardiac cycle. The magnitude of the vector field is represented by the color scale, while its direction is indicated by the arrows. The considered measurement unit is cm/s.
4 Simulation of biophysical phenomena in moving domains based on medical images

Figure 4.9: Screenshots displaying, on the left side of each panel, the difference between the WSS computed in the fixed domain simulation and in the moving domain simulation, on the surface of the domain. On the right side of each panel the WSS computed in the moving domain simulation is depicted. Both panels represent the solution at peak systole but the geometry is viewed from two different perspectives. The considered measurement unit is dyne/cm².

Figure 4.10: Geometry of the domain of interest at $t = 0ms$. 
Figure 4.11: Screenshots displaying the results of the simulation of the action potential propagation in a moving left ventricle at $t = 6, 18, 30, 42$ ms. The considered measurement unit is mV.
Figure 4.12: Zoom on the region of the left ventricle where the quality of the mesh degenerates abruptly at $t = 43$ ms.

Figure 4.13: Flow chart describing the proposed pipeline to validate the image-based moving domain approach, in the CFD case. The extension to the electro-cardiology case is straightforward.
Figure 4.14: Comparison of the fluid velocity computed by the fluid-structure interaction approach and the image-based motion technique, shown on a longitudinal section of the carotid domain. Each panel (identified by a black vertical line in the middle) displays on the left the magnitude of the difference field between the FSI velocity and the Image-Based (IB) motion velocity. On the right of each panel the IB motion velocity field is displayed. The first row corresponds to $t = 0.002$ s, the second row to $t = 0.004, 0.006$ s, while the third row to $t = 0.008$ s. The considered measurement unit is cm/s.
Figure 4.15: Comparison of the WSS computed by the fluid-structure interaction approach and the image-based motion technique on the lateral vessel wall. Each panel (identified by a black vertical line in the middle) displays on the left the magnitude of the difference field between the FSI WSS and the Image-Based (IB) motion WSS. On the right of each panel the IB motion WSS field is displayed. The first row corresponds to $t = 0.002$ s, the second row to $t = 0.004, 0.006$ s, while the third row to $t = 0.008$ s. The considered measurement unit is dyne/cm$^2$. 

123
5 An insight on algorithms and implementation details

In this Chapter we focus on the software implementation work that has been carried out to test and apply the numerical methods proposed in this thesis. In particular, in Section 5.1 we introduce the scientific computing software library that has been extensively used and developed during this work. We then describe in Section 5.2 the algorithm and the technical details that are relevant to test the method presented in Section 3.2. In Section 5.3 we explain the implementation of the model adaptivity algorithm presented in Section 3.3 and how the computational time has been reduced. In Section 5.4 we detail some aspects of the fluid-registration algorithm implementation (Section 4.2.1), highlighting the algorithms and data structures used, that feature a multi-purpose nature. Finally in Section 5.5 we describe the class hierarchy and methods that have been developed to exploit the available information about the domain motion.

5.1 The LifeV Library

LifeV [1] is a finite element (FE) software library for scientific computing, whose development started in 1999 as a joint work project involving Ecole Polytechnique Fédérale de Lausanne (CMCS) in Switzerland, Politecnico di Milano (MOX) in Italy, INRIA (REO) in France. Since 2008 Emory University (Department of Mathematics and Computer Science) in Georgia, U.S.A, has become part of the developers community. LifeV is a Free Software, subject to the LGPL license, and it is compilable on Unix-like systems. This software implements in C++ language algorithms and data structures to solve a numerical approximation of systems of partial differential equations, that arise e.g. from mathematical modeling applied to fluid-dynamics, fluid-structure interaction, flow in porous media, electrocardiology. LifeV has been employed in joint collaborations with medical and industrial partners (for instance Emory University, School of Medicine (GA - USA); Policlinico di Milano (Italy); Arena company) for research purposes.

LifeV is composed of a core part which provides an abstract framework for the implementation of Galerkin finite element methods, and of a set of solvers for specific applications. Although the library was originated as a serial software, recently the developers community started working on the porting of the whole library to parallel architectures.

---

1 On Windows systems this correspond to use Cygwin environment (http://www.cygwin.com/).
2 http://www.teamarena.com/en
5 An insight on algorithms and implementation details

We will focus hereafter on the serial version of the library, which was mainly used in this work. Nevertheless we mention that both the Monodomain and the Bidomain LifeV solvers have been ported to the parallel version during this thesis, showing good scalability properties in preliminary tests [95].

LifeV is able to solve both two and three dimensional problems. It interfaces with external libraries like Boost for containers, basic algorithms and smart pointers, BLAS and LAPACK for standard linear algebra tools, Aztec or Trilinos for matrix manipulation and linear systems solution.

The library is organized in layers (Figure 5.1). The most external layer is called test-

![Graphical representation of the layered structure of LifeV. The external layers depend on the internal ones.](image)

suite and collects the implementations of specific applications or simple tests to check the correct functioning of the library. The code requires user-defined data that specify the features of the test at hand. The testsuite is directly based on the part of the library called lifesolver, which contains data structures (organized in classes) and algorithms (that represent classes' methods) to solve a proper discretization of a specific PDEs system. The results obtained by the solvers can be postprocessed using methods provided in the section called lifefilters, which take care of writing on file the results, in a format compatible with common visualization software like Paraview, Ensight or Medit. Lifefilters contains also functionalities to read mesh files and to import previously stored solutions. A more internal layer is represented by lifefem, which contains the definition

---

3 Work done in collaboration with Dr. M. Perego (Emory University).
4 http://www.boost.org/
5 http://www.netlib.org/
7 http://trilinos.sandia.gov/
8 http://www.paraview.org/
9 http://www.ensight.com/
of various types of finite elements. Moreover, it includes the implementation of procedures to build the geometrical mapping and the quadrature rules for each of them and methods useful to assemble finite elements matrices and to define and apply boundary conditions. All the previously mentioned layers depend on lifemesh, containing data structures to handle computational meshes, and on lifealg, a collection of general purpose algorithms, such as non-linear search methods and wrappers for external libraries (in particular linear system solvers). The two most internal layers, used by all the other sections of the library, are lifearray, intended to provide classes and methods to easily manage matrices and arrays, and lifecore, containing basic functionalities like numerical type definitions, tools to interface with users data specifications and tools to help code debugging.

5.2 Monodomain preconditioner

In Section 3.2 we presented a preconditioner based on the Monodomain model. Optimality with respect to the mesh size has been proved and confirmed by numerical results. The implementation in LifeV has been based on Trilinos software. This choice is motivated essentially by two reasons. (a) Trilinos provides in BELOS package the Flexible GMRES algorithm, which is crucial for the performances of our preconditioner. (b) The parallel version of LifeV is currently based on Trilinos, so the porting to the parallel version would be easier, provided that the same interface is maintained. In this section we give details on the structure of the test code and the interface LifeV-Trilinos we developed. It is worth pointing out that another possible library for the same purpose is Aztec software. However, the numerical method representing the Aztec counterpart of FGMRES (which allows to use a different preconditioner in each iteration step), called GMRESR, is not documented in the software manual.

Some features of Trilinos We recall here a few Trilinos features that are important for the preconditioner implementation (see [2] for more details). The BELOS software package provides an implementation in C++ language of iterative methods for solving linear systems, together with generic interfaces (called SolverManager) to the available solvers and an abstract framework to define preconditioned or non-preconditioned linear systems to be solved. Among the features we mention the implementation of the block conjugate gradient, the block GMRES and the block flexible GMRES methods, which are exploited in our LifeV code.

Trilinos linear solvers are based on the use of proper specializations of Epetra Vector and Epetra RowMatrix classes, contained in the EPETRA package. EPETRA provides the fundamental construction routines and utility functions that are required for serial and parallel Trilinos linear algebra libraries, and consists in the underlying foundation for all Trilinos solvers. Epetra Vector represents a double precision vector class, support-

\[11\] http://trilinos.sandia.gov/
\[12\] http://acts.nersc.gov/aztec/
An insight on algorithms and implementation details

The construction and use of vectors in a parallel computing environment, while the Epetra_RowMatrix is a pure virtual class for sparse matrices. In the sequel we denote with Epetra_Matrix a proper specialization of this class.

To comply with BELOS linear problem interface, the code uses Trilinos RCP pointers, provided by TEUCHOS package, that are an implementation of C++ smart pointers.

LifeV implements wrapper classes of native EPETRA data structures, to simplify the interface between Trilinos functionalities and LifeV algorithms. In particular our code employs a template class EpetraMatrix, which is template with respect to the data type stored in the matrix and is a wrapper for a Trilinos sparse matrix class, and a class EpetraVector, a wrapper for a Trilinos vector container.

The implementation of the Monodomain preconditioned solver for the Bidomain problem was tested in a sequence of applications, whose general structure can be outlined as follows:

1. read the mesh file and input data;
2. instantiate objects and initialize containers for the unknowns of the problem;
3. assemble the finite elements matrices obtained after the space-time discretization of the problem (if time-invariant);
4. enter the temporal loop unless the last time step is reached:
   a) update time-dependent matrices and arrays (typically the right hand side of the system);
   b) solve the linear system;
   c) postprocess the solutions and write them on files;
   d) update previous time step unknowns.

Assembling of the Bidomain matrix Let us first introduce the outline of the assembling process for the discretized Bidomain system in LifeV (step 3). It is composed of a loop on the finite elements of the computational mesh, in which the contribution of each local element to the global matrix is computed, stored in an object of type ElemMat, and summed up in the global matrix.

```cpp
// i n i t i a l i z a t i o n o f the g l o b a l m a t r i x
* B_rcp *= 0.;
// l o o p on the e l e m e n t s o f the mesh
for (UInt i = 1; i <= mesh.numVolumes(); i++)
{
   // u p d a t e t h e s h a p e a n d b a s i s f u n c t i o n s q u a n t i t i e s
   // c o r r e s p o n d i n g to the current f i n i t e element
   fe.updateFirstDeriv(mesh.volumeList(i));
   elmatBido.zero();
   // c o m p u t e t h e m a s s m a t r i x c o n t r i b u t i o n to B_{(u,u)} block
   // a n d f i l l elmatBido with it
   mass(1./time_step, elmatBido, fe, 0, 0);
   // c o m p u t e t h e s t i f f n e s s m a t r i x c o n t r i b u t i o n s to every block
   // a n d f i l l elmatBido with them
   stiff(lambdaCoeff*sigmain/(chi*Cm), lambdaCoeff*sigmati/(chi*Cm));
```

127
5 An insight on algorithms and implementation details

```cpp
fibers, elmatBido, fe, dof, 0, 0);
stiff(lamda * Cm) - sigmale /((1 + lamda) * Cm),
lambda * Cm) - signale /((1 + lamda) * Cm),
fibers, elmatBido, fe, dof, 0.1);
stiff(sigmali, sigmait, fibers, elmatBido, fe, dof, 1.0);
stiff(sigmali+sigmale, sigmait+sigmate, fibers, elmatBido, fe, dof, 1.1);
//sum the local element contribution to the global matrix pointed by B_rcp
assemb_mat(*B_rcp, elmatBido, fe, dof, 0, 0);
assemb_mat(*B_rcp, elmatBido, fe, dof, 0, 1);
assemb_mat(*B_rcp, elmatBido, fe, dof, 1, 0);
assemb_mat(*B_rcp, elmatBido, fe, dof, 1, 1);
}
```

For details on each method cited in this code excerpt see [1] and [126], Chapter 11. We only mention that the method

```cpp
void stiff( Real sigmali, Real sigmait, const GenericVecHdl<Vector>& angles,
ElemMat& elmat, const CurrentFE& fe, const Dof& dof,
UInt iblock, UInt jblock);
```

is non standard since it implements both the construction of the conductivity tensors (2.8) and the associated stiffness matrices. The arguments \( \sigma_l \) and \( \sigma_t \) are the conductivity parameters of the tissue, angles is a vector containing the fibers direction, \( \text{fe} \) and \( \text{dof} \) are the data structures containing information on the considered finite element and its degrees of freedoms, while \( \text{iblock} \) and \( \text{jblock} \) select the global matrix block under examination.

We now consider point 4b assuming the previous steps accomplished. For a complete discussion on the structure of a finite element code, together with an example of implementation of a standard PDE test case in LifeV, we refer to [126], Chapter 11.

**Preconditioner Implementation**  
The solution of the linear system, obtained by the discretization of the Bidomain problem, is handled by an object of a class representing the linear system. The constructor for this object requires the matrix passed as an RCP pointer to an object of type Epetra_Mat, and the containers for the solution and the right hand side of the system, both passed as RCP pointers to Epetra_Vector objects.

```cpp
problemBidomain = rcp(new Belos::LinearProblem< double,MV,OP>( ABido_rcp, xBido_rcp, bBido_rcp ));
```

where MV is a type definition for Epetra_MultiVector and OP is a type definition for Epetra_Operator.

To set up a preconditioner for the linear problem just defined, the Belos LinearProblem class provides the methods

```cpp
void setLeftPrec(const RCP<const OP>& LP) { LP_ = LP; }
void setRightPrec(const RCP<const OP>& RP) { RP_ = RP; }
```

allowing to associate to the problem a left or a right preconditioner respectively. As mentioned in Section 3.2 Flexible GMRES algorithm requires a right preconditioner and the corresponding Epetra_Operator needs to be built.

The core of this work is the implementation of the Monodomain preconditioner. In the framework of Belos software package, this step is accomplished by defining a class
5 An insight on algorithms and implementation details

derived from the Epetra_OPERATOR base class, through the inheritance mechanism. For this reason, we implement the MonodomainOperator class, defined as

template <typename MatrixType>
class MonodomainOperator : public Epetra_Operator

providing a constructor

MonodomainOperator(RCP<MatrixType> const & mat11,
RCP<MatrixType> const & mat21,
RCP<MatrixType> const & mat22,
double const lambda,
EpetraMap & Map,
const Real & tol_data,
const UInt & recu);

We also need to overload the following method, required to apply the preconditioner:

int ApplyInverse(const Epetra_MultiVector & X, Epetra_MultiVector & Y) const;

This ApplyInverse method implements the solution of the Monodomain system and its efficiency has a crucial importance on the effectiveness of the operator, since this method is called by the iterative linear solver (FGMRES or GMRES for this test) at each iteration and is required to be faster than the application of other preconditioners. To this aim, the triangular Monodomain system is split into two linear sub-systems with matrices $B_{uu}$ and $B_{ue}$ respectively.

As a consequence, the MonodomainOperator class is designed to contain two LinearProblem and two SolverManager objects, associated with these systems. For each linear problem, an ILU preconditioner, implemented in Trilinos software, is instantiated and the Conjugate Gradient method is employed as a linear solver. We report hereafter some excerpts of the class definition and of the class constructor that show how one of the two sub-linear problems is set up. The snapshot of code referring to the other sub-problem is analogous.

// pointer to Ifpack preconditioner for system in $B_{uu}$
RCP<Ifpack_Preconditioner> PrecM11;

// pointer to Belos linear problem for system in $B_{uu}$
RCP<Belos::LinearProblem<double, MV, OP> > problem11;

// pointer to Belos solver manager for system in $B_{uu}$
RCP<Belos::SolverManager<double, MV, OP> > solver11;

// list to specify Ifpack parameters
ParameterList ifpackList;

// function class to define Ifpack preconditioner
IfpackFactory11;

// Epetra preconditioner operator for system in $B_{uu}$
RCP<Belos::EpetraPrecOp> belosPrecM11;

// definition of the preconditioner type and its parameters
std::string PrecType = "ILU";
int OverlapLevel = 1;
ifpackList.set("fact::drop_tolerance", 1e-9);
ifpackList.set("fact::level_of_fill", 1);
ifpackList.set("schwarz::combine_mode", "Add");

// initialization of the preconditioner
PrecM11 = rcp(Factory11.Create(PrecType, &M11_rcp, OverlapLevel));
assert(PrecM11 != Teuchos::null);
PrecM11->SetParameters(ifpackList);
PrecM11->Initialize();
5 An insight on algorithms and implementation details

// construction of the preconditioner
PrecM11->Compute();
belosPrecM11 = rcp( new Belos::EpetraPrecOp( PrecM11 ) );

// definition of the linear problem
problem11 = rcp( new Belos::LinearProblem<double,MV,OP>( M11.rcp, y1.rcp, x1.rcp ) );

// set up of the left ILU preconditioner
problem11->setLeftPrec( belosPrecM11 );

// update iteration-dependent matrix and rhs
bool set = problem11->setProblem();

// construction of the solver
solver11 = rcp( new Belos::BlockCGSolMgr<double,MV,OP>(problem11, rcp(&belosList, false)) );

The ApplyInverse method starts with the call to the solve method of the CG solver manager for the \( B_{uu} \) system, followed by the computation of the righthand side of the \( B_{ee} \) system. More precisely, by considering the second equation in (3.5), which is an approximation of the second equation in (3.6), and observing that \( \tilde{I}_{app} \) is zero in the considered test cases, we get an approximation for \( u_e \). In particular we obtain \( u_e = -\lambda u \), that can be used as an initial guess for the solution of \( B_{ee} \) system, to speed up the convergence. This is accomplished by initializing with this approximation the Epetra Vector that will be used as a container for the solution. The solution of the \( B_{ee} \) system can then be computed. We report hereafter an excerpt of this method.

// update iteration-dependent matrix and rhs
bool set = problem11->setProblem();

// solve the system in \( B_{(uu)} \)
Belos::ReturnType ret = solver11->solve();

// Check actual residuals (\( r = b - Ax \); the computation is split in steps)
bool badRes1 = false;
std::vector<double> actual_resids1( 1 );
std::vector<double> rhs_norm1( 1 );
EpetraVector resid1(map);
OPT::Apply(*M11.rcp, *y1.rcp, resid1.getEpetraVector());
MVT::MvAddMv( -1.0, resid1.getEpetraVector(), 1.0, *x1.rcp, resid1.getEpetraVector() );
MVT::MvNorm( resid1.getEpetraVector(), actual_resids1 );
MVT::MvNorm(*x1.rcp, rhs_norm1);
if ( actual_resids1[0] / rhs_norm1[0] > tol ) badRes1 = true;

if ( !Belos::Converged || badRes1 )
{
    std::cout << std::endl << "ERROR:...Belos did not converge!" << std::endl;
}
else std::cout << std::endl << "SUCCESS:...Belos converged!" << std::endl;

// Update rhs of the second sub-system with the solution of the first sub-system
M21.rcp->Multiply(0, *y1.rcp, *b2.rcp);

for (UInt i=0; i<dim; i++)
{
    (*x2.rcp)[i] = (*x2.rcp)[i] - (*b2.rcp)[i];
    // Provide an initial guess for the second sub-system, motivated by the physics of the problem
    (*y2.rcp)[i] = -lambda*coeff(*y1.rcp)[i];
}
Once the definition of the MonodomainPreconditioner class is available, an object of this class can be instantiated:

```cpp
MonodomainPrec_rcp = rcp(new MonodomainOperator<Epetra_Matrix>(
    M11_rcp, M21_rcp, M22_rcp, lambda, map, tol_Mprec, 1));
belosPrec_rcp = rcp(new Belos::EpetraPrecOp(MonoPrec_rcp));
```

having denoted with \(M_{ij}\) an RCP pointer to block \(i,j\) of the Bidomain matrix, with \(\lambda\) the Monodomain parameter, with \(\text{map}\) the Epetra Map (a relevant parameter only on parallel architectures), with \(\text{tol}_M\text{prec}\) the tolerance to evaluate the convergence of the CG algorithms mentioned before. Then, similarly to what is done in the MonodomainPreconditioner constructor for \(B_{uu}\) and \(B_{ee}\) system, the solution of the Bidomain system is computed, using a Flexible GMRES solver. We report only the snapshot of code that differs substantially from the previous description, in particular the setting of the user-defined \(\text{belosPrec}_{\text{rcp}}\) preconditioner as a right preconditioner and the instantiation of a Flexible GMRES solver manager.

```cpp
// set the right preconditioner for the problem Bidomain
problemBidomain->setRightPrec( belosPrec_rcp );
```

The solver manager \(\text{solverBidomain}\) is instantiated by choosing the numerical method for solving the system and by defining the linear system under consideration. This information is stored in the linear problem \(\text{problemBidomain}\), which also contains the information on the preconditioner, set using the method \(\text{setRightPrec}\). The \(\text{ApplyInverse}\) method will be then called in the the FGMRES algorithm, to solve the preconditioner system.

The solution of the Bidomain linear system concludes step 4b of the algorithm. The implementation of the remaining steps is not detailed here, since it involves pretty standard procedures.

### 5.3 Model Adaptivity

In Section 3.3 a model adaptivity algorithm is presented, to reduce the computational cost of the action potential propagation. We detail in this Section the implementation of the algorithm in LifeV, highlighting the expedients used to achieve an effective computation.

The implementation of this test case in LifeV follows again the same general structure presented in Section 5.2. The main peculiar features of the algorithm, relatively to the
implementation, refer to steps 4a and 3 with the evaluation of the model error estimator (3.38) and the assembling of the Hybridomain problem matrix. It is worth noting that step 4b still involves the Monodomain preconditioner described in the previous Section. The other steps are still standard.

Let us now consider how the Hybridomain finite element matrix is built. If the coefficients of the mass and stiffness matrices do not change in time during the simulation, this assembling procedure described in Section 5.2 that is quite expensive in terms of computational time, can be performed offline. In other words, it is executed only once in the whole simulation. The part of the code that is placed inside the time loop and therefore repeated at each time step is referred to as online. While for the Bidomain and Monodomain simulations all the assembling phase can be performed offline, the adaptivity algorithm, described in Section 3.3.3, is based on the online updating of the upper right block of the finite element matrix. As a consequence, in the Hybridomain case the online phase would be more expensive than in the standard Bidomain one, influencing the effectiveness of the strategy.

To reduce the computational costs, we proceed as follows. For each element of the grid, the local matrix associated with the upper right block is pre-computed offline. At this stage these contributions are stored in a vector of ElemMat, without being assembled in the global matrix. Moreover we completely assemble in the global matrix all the time-invariant matrix blocks (upper left block and both blocks on the lower row). The vector of ElemMat, named elmat_B_ue and storing the local contributions to be possibly assembled, is built as follows:

```c
// computation of the local contribution of the considered FE to B_{ue}
stiff(lambdacoef*sigmali/(chi*Cm), lambdacoef*sigmati/(chi*Cm),
     fibers, elmat_B_ue[i-1], fe, dof, 0, 1);
stiff(-sigmali/((1+lambda)*chi*Cm), -sigmati/((1+lambda)*chi*Cm),
     fibers, elmat_B_ue[i-1], fe, dof, 0, 1);
```

The same approach is used to compute the discretization of the estimators and the error, storing the contribution of each local matrix in a vector to be used in the time step loop. For instance, to compute the stiffness matrix involving the tensor $D_s$, we call

```c
stiff_upperbound(sigmale, sigmate, sigmali, sigmati,
     fibers, elmatUpBoundVec[i-1], fe, dof, 0, 0);
```

which is similar to the Bidomain stiffness method previously mentioned, differing only for the tensor computed, that in this case reads

$$D_s^T(D_i^{-1} + D_e^{-1})D_s.$$ 

Let us focus now on the online phase, when the upper right block of the Hybridomain matrix is assembled. At each time step and for each element, the local error indicator (3.38) is checked. If it is greater than the chosen threshold, then the current element is marked as “Bidomain element” and its contribution will be summed to the upper right block of the global Hybridomain matrix. To compute the local error indicator (3.38) the following method has been implemented in the test solver:
It takes as arguments the considered finite element \(fe\), its index in the elements list \(i\), and the solution vector \(ue\_stim\) used to evaluate the estimator. Using this information and the \(elmat\_Up\_Bound\_Vec\) vector built offline, the local contribution to the error indicator is computed.

From equations (3.34) and (3.37) it appears that the global error upper and lower bounds involve a volume integral defined only on the region of the domain labelled as \(\Omega_M\). For this reason, the contribution to the global error indicator is considered only if the local error indicator is smaller than the threshold. On the other hand the norm of the error is defined by integrals on both the partitions.

Once all the assembling procedure is completed, the finite element matrix pointed to by \(H_{rcp}\) is ready to be passed as an argument to the linear solver described in Section 5.2.

5.4 Fluid-Registration algorithm

We describe in this Section the LifeV code developed to test the behaviour of the finite element fluid registration algorithm, presented in Section 4.2.1. In particular we outline the structure of the test and we provide more details on the methods implemented ad hoc for this test. The fluid registration algorithm implemented is designed to work both in two and three dimensions. The only difference between the two cases lays in the implementation of the method \(\text{is.in}\), used to check if a given position in the space is inside a finite element. In 2D this step takes advantage of the structured computational grid used and is therefore faster. Hereafter we consider only the 3D case, that is more general, since it does not rely upon a specific structure of the mesh. The method presented is actually independent of the space dimension and the finite element used.

\[13\] To compute the lower bound an analogous method is implemented.
5 An insight on algorithms and implementation details

The fluid-registration procedure associated to the solution of system (4.6) reads as follows.

1. read mesh file, input data and images to be registered
2. instantiate objects and initialize containers for the unknowns of the problem
3. assemble the time-invariant finite element matrices
4. enter the loop unless convergence is reached
   a) update time-dependent matrices and solutions
   b) evaluate the right hand side terms on the transformed grid
   c) solve equation (4.7) for $u$
   d) solve equation (4.8) for $d$
   e) postprocessing

The most interesting step of the algorithm is 4b. We split it into four substeps:

1. computation of the new position of the grid points, according to the displacement field obtained at the previous iteration;
2. computation of the gradient of the template function $T$;
3. for each new position, identification of the element in the initial grid enclosing the node;
4. evaluation of the right hand side of equation (4.7) on the deformed grid.

This last substep is performed by interpolation of the template intensity function and of its gradient in the element identified at substep 3.

The computation of the gradient is performed by looking for an $L^2$ approximation of the considered field ($\nabla T$ in this case), as done for the computation of the wall shear stress in Section 4.3. The algorithm was already implemented in LifeV and is described in [114], Chapter 3.4.

Substeps 3 and 4 make use of a class called PointLocator that we added to the library. In particular we mention the following methods.

```cpp
template <typename Mesh>
ID locateNearestPoint(const PointType& q, Real& dist_tol) const;
ID findEleID(PointType& point, Real tol = 0.) const;
bool is_in(const PointType& point, ID eleID, Real tol = 0.) const;
Vector evaluate(Vector& unknown_field, Vector& disp_field, CurrentFE& fe, Real tol = 0.);
```

The first represents a wrapper for an external library, named ANN\(^1\) which implements a fast approximate nearest neighbor searching. More precisely, the class constructor builds a list of data points $p_i$, $i \in 1...N$. Given a query point $q$, this method returns the point in the data set that is the nearest to $q$, checking their distance in comparison with a tolerance given as input argument to the method. The main lines of this wrapper are

\[^1\]http://www.cs.umd.edu/ mount/ANN/
that correspond to the creation of the optimized search structure and the call to the ANN search method.

The goal of findEleID method is to find the grid element (if any) which contains a given point of the space. The algorithm developed is based on a preliminary step (performed in the constructor of PointLocator class) that fills the data structure.

```
std::vector<std::vector<ID> > patch_vec;
```

The latter is a vector whose length corresponds to the number of nodes. Each entry is in turn a vector pointing to the list of elements of the grid sharing the node (patch). To clarify this step let us make a 2D example with a simple triangles grid, shown in Figure 5.2, where the letters stand for the elements labels (integer numbers in the code) and the numbers represent the nodes labels. In this case patch_vec would be

![Figure 5.2: Example of 2D triangular grid. The numbers correspond to the vertices labels while the letters stand for the elements labels.](image-url)
5 An insight on algorithms and implementation details

\[
\text{patch_vec} = \begin{bmatrix}
  a, i \\
  a, b \\
  a, b, c, h, i \\
  b, c, d \\
  d, e \\
  c, d, e, f, g, h \\
  e, f \\
  f, g \\
  g, h, i 
\end{bmatrix}
\] (5.1)

Thanks to the information stored in this vector, the element searching algorithm, for a given point \( q \) is the following:

1. search the nearest neighbor of \( q \) among all the mesh nodes
2. scroll the sub-vector of \( \text{patch_vec} \) corresponding to the found mesh node, selecting the first available element
3. check if \( q \) belongs to the selected element
   a) if yes, then return the element identifier
   b) else, proceed to the next element in the \( \text{patch_vec} \) vector and go to step 3.

If all the elements in the sub-vector have been checked, then return 0.

A return value of zero corresponds to the case of a point \( q \) which has been moved out of the mesh boundaries. However in the fluid-registration problem, since a homogeneous Dirichlet boundary condition is prescribed to the velocity \( u \) on the boundary of the image, this case would correspond to a degeneration of the transformed grid, causing the algorithm to end. However, provided that the advancing step is small enough, this situation is unlikely to happen. We report hereafter the core part of the \text{findEleID} method:

```cpp
nearest_ID = locateNearestPoint(point, d);
bool found(0);
for (UInt i=0;i<patch_vec[nearest_ID].size(); ++i)
{
    if (is_in(point, mesh.element(patch_vec[nearest_ID][i]).id(), tol))
    {
        found = 1;
        returnvar = patch_vec[nearest_ID][i];
        break;
    }
    else found=0;
}
if (found==1) return returnvar;
else
{
    std::cout << "ERROR: no eleID found for point (" << point.x() << ", " << point.y() << ", " << point.z() << ")" << std::endl;
    return 0;
}
```

136
As mentioned at the beginning of this Section, the method is devoted to check if a given input point lays inside the finite element with identifier eleID. The algorithm we present can be applied to tetrahedral elements. Its generalization to 2D elements or different 3D geometrical elements is straightforward.

The basic idea of the algorithm is the following: each face of the tetrahedral element defines a plane, described by its out-going normal. Three vertices of the tetrahedral finite element belong to the plane by construction, while the fourth vertex is external to the plane. For each plane, we check whether point and the fourth vertex lay in the same semi-space. If this check gives a positive result for all the four faces of the element, then the considered point is inside the finite element, otherwise it lays outside the element. We report the main part of the code implementing this method.

```cpp
// fe_vertices is a ublas vector containing the coordinates of all the vertices of the element
UInt success=0;
for (UInt i=0; i<nDimensions+1; i++)
{
    // store the vectors starting from a chosen vertex (namely fe_vertices[i%(nDimensions+1)]) and pointing towards the other vertices and towards point
    v1=fe_vertices[(i+1)%(nDimensions+1)]-fe_vertices[i%(nDimensions+1)];
    v2=fe_vertices[(i+2)%(nDimensions+1)]-fe_vertices[i%(nDimensions+1)];
    v3=fe_vertices[(i+3)%(nDimensions+1)]-fe_vertices[i%(nDimensions+1)];
    v4=point-fe_vertices[i%(nDimensions+1)];
    // compute the normal to the face, defined by the vector product v1xv2
    face_normal=vector_product(v1,v2);
    // compute the scalar product n\cdot v3
    r1=scalar_product(face_normal,v3);
    // compute the scalar product n\cdot v4
    r2=scalar_product(face_normal,v4);
    // check is point and v3 and v4 lay in the same semi-space
    if (r1*r2>=0-tol) success++;
}
if (success==(nDimensions+1)) return 1;
else return 0;
```
linear system. By linear interpolation, we get the value of the field in $p$:

$$u(p) = \sum_{j=1}^{n+1} \phi_j u(v_j),$$

where the value of $u$ on the element vertices is known.

Remark 1  It is worth pointing out that the method findEleID is a general purpose facility. For instance, we used it in Section 3.3.4 for the identification of the scar reconstructed from medical images. In this test case we introduce a mesh for the scar, independent of the computational mesh of the heart. To identify the region of the heart affected by the scar, each point of the heart mesh is tested by the findEleID method to check if it belongs to the scar geometry. Points of the heart mesh belonging to the scar geometry are marked and their local conductivity is changed accordingly to the pathological value.

Remark 2  The evaluate method has a general purpose too. In particular it helps in applying the map given by the registration algorithm to transform a set of points in the image space. Actually, these points may not belong to the set of nodes of the initial mesh, on which the map is defined. For example, in the test presented in Section 4.2.1, the contour of a myocardium slice is segmented, obtaining a set of points which can have a higher spatial resolution than the grid discretization. To move this contour according to the displacement field computed by the fluid-registration algorithm (hence defined on the mesh grid), an interpolation procedure is required. This step can be accomplished by the evaluate method.

5.5 Moving domain simulations

In this section we describe the algorithm used to perform simulations in image-based moving domains, introduced in Section 4.3, that can be summarized as follows:

1. read mesh file and input data
2. instantiate objects and initialize containers for the unknowns of the problem
3. assemble the (time-invariant) finite element matrices obtained after the space-time discretization of the problem
4. enter the temporal loop unless the last time step is reached
   a) import and process the motion data related to the current step
   b) update time-dependent matrices and arrays, according to the current displacement and computed solution
   c) solve the linear system
5 An insight on algorithms and implementation details

d) postprocess the solutions and save them on files
e) update previous-time step unknowns

The algorithm presented is valid to perform both the fluid dynamics and the electrocardiology simulations in image-based (IB) moving domain. In particular the implementation of step 4a is the same for both the applications, while the actual application of the other steps of the algorithm depends on the nature of the problem.

To simplify the discussion, we focus here only on the fluid-dynamic case, reporting at the end of the description, the main differences with respect to the electrocardiology case.

The implementation of the algorithm is strongly based on the NavierStokesAleSolver class, already present in LifeV.\textsuperscript{15} The goal of the original implementation was to provide a solver for the Navier-Stokes equations in ALE formulation, to be coupled with a solver for the solid mechanics of the vessel wall. This class has been slightly modified to include the information on the domain displacement coming from user’s data, by the introduction of step 4a. The corresponding method in the solver class is

\begin{lstlisting}[language=C++]
void getDisplacementFromData( Real& t, Real dt );
\end{lstlisting}

The interface with displacement data is handled by an instance of

\begin{lstlisting}[language=C++]
template<typename Mesh > class ImposedMovement
\end{lstlisting}

ImposedMovement is an abstract class with two specializations called AnalyticalMovement and MovementFromFile.\textsuperscript{16} The first specialization is used to simulate the radial motion of an idealized cylindrical vessel, and it is useful when limited information is available on the geometry and its motion, requiring simplifying hypotheses to proceed. In this Section we discuss only the MovementFromFile specialization, since it is the one employed in the numerical tests of Section 4.3.

The most relevant methods of this class are the private data importer method

\begin{lstlisting}[language=C++]
typename ImposedMovement<Mesh>::PointVectorType
getPointsFromFile( const std::string& filename, std::vector<ID>& boundaryIDs );
\end{lstlisting}

and the overloading of public base class’ virtual methods:

\begin{lstlisting}[language=C++]
virtual void computeDisp( const Real& t, const Real& dt, bool& new_disp );
virtual void fillAllBdDisp( Vector& all_bd_disp );
\end{lstlisting}

The first method is able to read the current positions of the nodes of the boundary surface of the geometry and to associate them to the corresponding nodes of the volume mesh. Various file formats can be read.

The two overloading methods select the data file to be imported, depending on the current time t, the time step of the simulation dt and the time interval between two subsequent image frames frame_step:

\begin{lstlisting}[language=C++]
UInt frame_next_idx=(static_cast<int>((t-dt/2)/frame_step))+1;
\end{lstlisting}

\textsuperscript{15} The original code has been implemented by Dr. M. A. Fernandez, INRIA (France), in 2005.
\textsuperscript{16} This inheritance tree has been implemented in collaboration with Dr. T. Passerini, Emory University, author of the AnalyticalMovement specialization.
An insight on algorithms and implementation details

Then the displacement with respect to the previous time step of the simulation is computed and the information is stored in the boundary positions of a nodal vector.

These methods are called from the `getDisplacementFromData` method of `NavierStokesAleSolver` class, which is also in charge of calling the computation of the harmonic extension of the boundary displacement, managed by an instance of the class

```cpp
template <typename Mesh>
class HarmonicExtension
```

already present in LifeV.\(^{17}\)

After step 4a is performed, the norm of the computed displacement is evaluated and, if it is greater than a chosen threshold, all the finite elements matrices of the problem are re-computed and the algorithm proceeds. In the fluid dynamics case, the updated finite element matrices will include the latest value of the fluid and the mesh velocity field. On the other hand, in the electrocardiology implementation, the online update of the finite element matrices, due to the movement of the mesh, requires the transformation of the cardiac fibers vector field. More precisely from the displacement field, the gradient of the deformation can be computed and used to transform the fibers vector field. This procedure is performed by the method

```cpp
void update(const CurrentFE& fe, const Vector& d);
```

that we implemented in LifeV, for the class `ConductivityTensor`. This class stores all the information regarding the local conductivities and the fibers direction. The `update` method compute for each quadrature node of each finite element `fe`, the new the local fiber direction, according to (2.21), on the basis of the displacement `d`.

---

\(^{17}\)The HarmonicExtension method has been implemented by Dr. M. A. Fernandez, INRIA (France), in 2002.
6 Conclusions

The research carried out in this thesis has been mainly devoted to develop techniques and to conceive practical procedures for numerical simulation of biophysical phenomena in the cardiovascular system.

The possibility of applying these findings to medical applications has been the driving force of the work. Not only established knowledge retrieved from the literature, but also the personal discussion with medical doctors and researchers motivated this activity, highlighting the importance of a tight collaboration of scientists with different extraction on the study of physiology and pathology of the human system.

After the introduction to the medical context (Chapter 1), we focused on the most common models available in the literature to describe the action potential propagation in the myocardium, in fixed and moving frameworks, and to take into account the mechanics of the tissue (Chapter 2). We also reported the fluid-mechanics models employed to simulate blood flow in large arteries and to devise a specific type of image registration algorithm described in Section 4.2.1. Then we briefly presented a common model used to describe the arterial wall mechanics.

Chapter 3 is devoted to numerical methods for simulating the action potential propagation in the ventricles wall. After an overview on the strategies and software packages available in the literature, we concentrated on the contributions developed in this work. In particular we devised an ad-hoc preconditioning strategy for the Bidomain problem, which is based on a proper reformulation of the standard system and on the use of a simplified problem (the so-called Extended Monodomain model) as a preconditioner. We proved that this is an optimal preconditioner, since the condition number of the preconditioned operator is limited independently of the space discretization parameter. We also showed some numerical evidences on 3D computations that confirm this theoretical result, in particular comparing the Monodomain preconditioner strategy with a common standard approach. The described algorithm has been implemented as part of the serial version of the software library LifeV. A straightforward extension of the present work consists in porting the code to a parallel implementation and test its scalability on parallel architectures, where we expect a remarkable reduction of the computational effort.

One of the advantage of this ad-hoc numerical method is that it can be easily combined with other strategies, to further reduce the computational time. As a matter of fact the Monodomain preconditioner has been coupled with a model adaptivity strategy. It is based on the definition of a model error estimator which controls the difference between the solution of the Bidomain and the solution of the so-called Hybridomain model. The latter is a novel model to describe the action potential propagation and corresponds to solving the Bidomain model in a (small) partition of the computational
6 Conclusions

domain and the Monodomain one in the remaining part. On the basis of the model error estimator, an adaptive algorithm is proposed, able to select automatically the partition of the domain and to simulate the potential propagation saving computational time and maintaining a good accuracy of the results. Again, this strategy has been tested so far in the serial version of LifeV library and its porting to parallel implementation will be object of future developments. Moreover, the adaptive algorithm will be also tested in a different framework: instead of formulating a “monolithic” problem containing both the Bidomain and the Monodomain part, as in the case of the Hybridomain model, we will couple the model adaptivity strategy with a domain decomposition approach, where the two subproblems are solved separately. This approach seems to be promising since it would avoid the use of an extended version of the Monodomain, which is more expensive to be solved than the original Monodomain formulation. However the convergence of the domain decomposition algorithm becomes in this case crucial for the effectiveness of the overall strategy.

In Chapter 4 we focused on the motion of the biological structures in regions where phenomena of interest, as blood flow and potential propagation, take place. We presented a possible pipeline to extract information on the motion from medical images and to include it in the numerical simulations. We described the steps that compose the pipeline, with particular attention to an image registration algorithm which seems to be suitable for tracking large deformations, occurring in the heart. Preliminary results of the application of this technique are presented and will be consolidated in the future development of this work. We then applied the image-based motion strategy to the modeling of blood flow in a patient-specific aortic arch geometry, comparing the results with a fixed-domain simulation. Finally we proposed a validation of the approach, through a comparison with the results of a more standard fluid-structure interaction algorithm. The technique is promising since the differences between the two computed solution fields are very small and the image-based motion numerical simulations require less computational effort than standard FSI strategies. On the other hand, we point out that this approach consists in a simplified modeling of the coupled fluid-structure problem, since only the effect of the structure on the fluid is taken into account, while the fluid feed-back on the structure is neglected. However the great advantage of this approach is that a tight link with the actual motion of the biological structure is maintained, without unaffordable computational efforts. This work can be complemented by performing a more extensive set of numerical experiments and investigating the dependence of the results on parameters like those defining the space and time discretization.

The same pipeline can be applied in principle to perform electrocardiology simulations in a moving heart. However the large deformations occurring in the heart pose some technical problems in the motion tracking procedure. The future development of this work consists in including, in the imaged-based motion modeling pipeline, an ad-hoc registration technique able to overcome this issue. Also in this case, the image-based motion strategy needs to be validated with respect to more standard electromechanical coupling algorithms.

To conclude, the most relevant aspects of the implementation of the algorithms in LifeV library are collected in Chapter 5. It is indeed crucial that the modeling techniques
6 Conclusions

devised for studying the human system are implemented in effective, robust and usable software. This is the most natural tool for the communication and collaboration of different scientific communities, and is likely to promote new, effective and more quantitative methods in the clinical practice.
A List of parameters

A.1 Tissue models

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\sigma_l^i$</td>
<td>$3 \times 10^{-3} , \Omega^{-1} , cm^{-1}$</td>
<td>Intracellular conductivity along fibers [33]</td>
</tr>
<tr>
<td>$\sigma_l^e$</td>
<td>$2 \times 10^{-3} , \Omega^{-1} , cm^{-1}$</td>
<td>Extracellular conductivity along fibers [33]</td>
</tr>
<tr>
<td>$\sigma_t^i$</td>
<td>$3.1525 \times 10^{-4} , \Omega^{-1} , cm^{-1}$</td>
<td>Intracellular conductivity transversal to the fibers [33]</td>
</tr>
<tr>
<td>$\sigma_t^e$</td>
<td>$1.3514 \times 10^{-3} , \Omega^{-1} , cm^{-1}$</td>
<td>Extracellular conductivity transversal to the fibers [33]</td>
</tr>
<tr>
<td>$\chi$</td>
<td>$1 \times 10^3 , cm^{-1}$</td>
<td>Membrane surface to volume ratio [33]</td>
</tr>
<tr>
<td>$C_m$</td>
<td>$1 \times 10^{-3} , mF , cm^{-2}$</td>
<td>Membrane capacitance [33]</td>
</tr>
</tbody>
</table>

Table A.1: Conductivities values and parameters used in the numerical simulation of the Bidomain (2.12) and Monodomain models (2.19). After [117].

A.2 Ionic models

A.2.1 Rogers-McCulloch model

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$G$</td>
<td>$1.5 , \Omega^{-1} , cm^{-2}$</td>
<td>[33]</td>
</tr>
<tr>
<td>$\eta_1$</td>
<td>$4.4 , \Omega^{-1} , cm^{-2}$</td>
<td>[33]</td>
</tr>
<tr>
<td>$\eta_2$</td>
<td>$0.012$</td>
<td>[33]</td>
</tr>
<tr>
<td>$\eta_3$</td>
<td>$1$</td>
<td>[33]</td>
</tr>
<tr>
<td>$v_{th}$</td>
<td>$13 , mV$</td>
<td>[33]</td>
</tr>
<tr>
<td>$v_p$</td>
<td>$100 , mV$</td>
<td>[33]</td>
</tr>
</tbody>
</table>

Table A.2: Values of the parameters of the Rogers-McCulloch model (2.3) used in the present work.
A List of parameters

A.2.2 Luo-Rudy model

We report here after the detailed version of the Luo-Rudy phase I model [91], including the model parameters used in the simulations.

\[ w = [h \ j \ m \ d \ f \ X]^T, \]
\[ a = -[(\alpha_h + \beta_h), (\alpha_j + \beta_j), (\alpha_m + \beta_m), (\alpha_d + \beta_d), (\alpha_f + \beta_f), (\alpha_X + \beta_X)]^T, \]
\[ b = [\alpha_h \ \alpha_j \ \alpha_m \ \alpha_d \ \alpha_f \ \alpha_X]^T, \]
\[ c = [Ca], \ g = -10^{-4}I_{si} + 0.07(10^{-4} - Ca), \]
\[ I_{ion} = I_{K1} + I_{Kp} + I_b + I_K + I_{Na} + I_{si}, \]
\[ u_0 = -84mV, \ c_0 = 2e - 4mM, \ w_0 = [1 \ 1 \ 0 \ 0 \ 1 \ 0]^T, \]
\[ I_{Na} = 23m^3 h j (u - v_{Na}), \quad v_{Na} = 54.4mV \]

\[ \alpha_h = \begin{cases} 
0 & u \geq -40 \\
0.135e^{-\frac{80+u}{60}} & u < -40 
\end{cases} \]

\[ \beta_h = \begin{cases} 
0.13 \left(1 + e^{-\frac{u+10.66}{11.1}}\right) & u \geq -40 \\
3.56e^{0.079u} + 3.1 \cdot 10^5 e^{0.35u} & u < -40 
\end{cases} \]

\[ \alpha_j = \begin{cases} 
0 & u \geq -40 \\
\frac{-2.535 \cdot 10^{-7}u}{1 + e^{0.1(u+32)}} & u \leq -40 \\
\frac{3.76 \cdot 10^{15} e^{0.1052u}}{1 + e^{-0.01052u}} & u < -40 
\end{cases} \]

\[ \beta_j = \begin{cases} 
\frac{u + 37.78}{1 + e^{0.01(u+5)}} & u \geq -40 \\
0.121e^{-0.01052u} & u < -40 
\end{cases} \]

\[ \alpha_m = \frac{0.32 \cdot 10^{-5} u}{1 - e^{-0.1(u+17.13)}} \]

\[ \beta_m = 0.08e^{-\frac{u}{11}} \]

\[ \alpha_d = 0.095 \frac{e^{-0.001(u-5)}}{1 + e^{-0.072(u-5)}}, \]

\[ \beta_d = 0.07 \frac{e^{-0.017(u+44)}}{1 + e^{0.05(u+44)}} \]

\[ \alpha_f = 0.012 \frac{e^{-0.008(u+28)}}{1 + e^{0.15(u+28)}}, \]

\[ \beta_f = 0.0065 \frac{e^{-0.02(u+30)}}{1 + e^{-0.2(u+30)}}, \]

\[ \alpha_X = 0.0005 \frac{e^{0.083(u+50)}}{1 + e^{0.057(u+50)}}, \]

\[ \beta_X = 0.0013 \frac{e^{-0.06(u+20)}}{1 + e^{-0.04(u+20)}}, \]
A List of parameters

\[ I_{si} = 0.09 \, d \, f(u - v_{si}), \]
\[ v_{si} = 7.7 - 13.0287 \ln(Ca), \]
\[ I_K = \overline{g}_K X_i(u - v_K), \quad v_K = -77.01 \, mV, \quad \overline{g}_K = 0.282 \sqrt{K_o \frac{1}{5.4}}, \quad K_o = 5.4 \, mM, \]
\[ X_i = \begin{cases} 
2.837 \frac{e^{0.04(u+77)} - 1}{(u + 77)e^{0.04(u+35)}} & u > -100.05, \\
1 & u \leq -100.05,
\end{cases} \]
\[ I_{K1} = \overline{g}_K X_1 \frac{\alpha_K}{\alpha_K + \beta_K} (u - v_{K1}), \quad v_{K1} = -87.26 \, mV, \quad \overline{g}_{K1} = 0.282 \sqrt{\frac{K_o}{5.4}}, \]
\[ \alpha_{K1} = \frac{1}{1 + e^{0.2385(u-v_{K1} - 59.215)}}, \]
\[ \beta_{K1} = \frac{0.49124 e^{0.08032(u-v_{K1}+5.476)} + e^{0.06175(u-v_{K1}-504.31)}}{1 + e^{-0.5143(u-v_{K1}+4.753)}}, \]
\[ I_{Kp} = 0.0183 K_p (u - v_{Kp}), \quad v_{Kp} = v_{K1}, \]
\[ K_p = \frac{1}{1 + e^{\frac{u-108.5}{5.95}}}, \]
\[ I_b = 0.03921 (u + 59.87), \]
\[ I_{K1(T)} = I_{K1} + I_{Kp} + I_b \]
Acknowledgements

I warmly and sincerely thank my supervisor for his constant support and trust, which have always motivated me to do my best.

I am grateful to all the researchers I collaborate with, for having contributed to set up a unique and valuable work scenario, where personal affinity fuels also the scientific production.

I would like also to thank the institutions that hosted me during my ph. D. program: the Department of Mathematics of Politecnico di Milano and in particular the MOX lab for having initiated me to this career and for having offered me the possibility of working in an excellent environment; the Department of Mathematics and Computer Science of Emory University for having welcomed me in a context full of challenges and rich of interesting stimuli.
Bibliography


Bibliography


150
Bibliography


Bibliography


Bibliography


Bibliography


Bibliography


Bibliography


