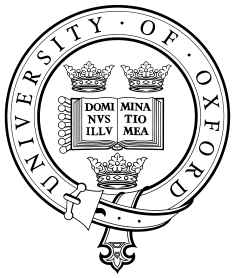
An open source HPC-enabled model of cardiac defibrillation of the human heart

Miguel Óscar Bernabeu Llinares
Linacre College



Computational Biology Group
Department of Computer Science
University of Oxford

Trinity Term 2011

This Thesis is submitted to the University of Oxford for the degree of Doctor of Philosophy. This Thesis is entirely my own work, and, except where otherwise indicated, describes my own research.

Miguel Óscar Bernabeu Llinares
Linacre College

Doctor of Philosophy
Trinity Term 2011

An open source HPC-enabled model of cardiac defibrillation of the human heart

Abstract

Sudden cardiac death following cardiac arrest is a major killer in the industrialised world. The leading cause of sudden cardiac death are disturbances in the normal electrical activation of cardiac tissue, known as cardiac arrhythmia, which severely compromise the ability of the heart to fulfill the body's demand of oxygen. Ventricular fibrillation (VF) is the most deadly form of cardiac arrhythmia. Furthermore, electrical defibrillation through the application of strong electric shocks to the heart is the only effective therapy against VF. Over the past decades, a large body of research has dealt with the study of the mechanisms underpinning the success or failure of defibrillation shocks. The main mechanism of shock failure involves shocks terminating VF but leaving the appropriate electrical substrate for new VF episodes to rapidly follow (i.e. shock-induced arrhythmogenesis).

A large number of models have been developed for the in silico study of shock-induced arrhythmogenesis, ranging from single cell models to three-dimensional ventricular models of small mammalian species. However, no extrapolation of the results obtained in the aforementioned studies has been done in human models of ventricular electrophysiology. The main reason is the large computational requirements associated with the solution of the bidomain equations of cardiac electrophysiology over large anatomically-accurate geometrical models including representation of fibre orientation and transmembrane kinetics.

In this Thesis we develop simulation technology for the study of cardiac defibrillation in the human heart in the framework of the open source simulation environment Chaste. The advances include the development of novel computational and numerical techniques for the solution of the bidomain equations in large-scale high performance computing resources. More specifically, we have considered the implementation of effective domain decomposition, the development of new numerical techniques for the reduction of communication in Chaste's finite element method (FEM) solver, and the development of mesh-independent preconditioners for the solution of the linear system arising from the FEM discretisation of the bidomain equations.

The developments presented in this Thesis have brought Chaste to the level of performance and functionality required to perform bidomain simulations with large three-dimensional cardiac geometries made of tens of millions of nodes and including accurate representation of fibre orientation and membrane kinetics. This advances have enabled the in silico study of shock-induced arrhythmogenesis for the first time in the human heart, therefore bridging an important gap in the field of cardiac defibrillation research.

To anybody who has ever taught me anything.

Per damunt de tot als meus pares, per ensenyar-me la ferramenta que ho ha fet tot possible: l'esforç.

Acknowledgements

This Thesis is the culmination of many years of work and it would not have been possible without the help of many people.

I would like to thank first my supervisors, Drs Joe Pitt-Francis, Blanca Rodríguez, and David Kay, for the opportunity of coming to Oxford to read for my doctorate and for their knowledgeable advice. It has been an amazing journey and I will be forever in debt to them for exposing me to so many different and fascinating research topics. Professor David Gavaghan also co-supervised my doctorate and provided diligent advice in the preparation of this manuscript. Professor Kevin Burrage played an important role providing external criticism and helping to shape the project.

My biggest thank you goes to you Liren for your love and care. We have walked this sometimes tough way together and you have given me the peace I needed to keep sane. A big portion of the work in this Thesis has been fueled by your amazing cooking as well.

I would also like to thank all the guys working on Chaste, currently and over the past years. This Thesis would not have been posible without their wise advice and constant attention to all the chaos I created while trying "new things".

Next, I would like to thank my family and friends in Spain who made me feel so very welcome in each of my trips back and also paid so many unforgettable visits to Oxford.

I would also like to thank my former colleagues at the Universitat Politècnica de València who taught me some of the skills that I have used in this work.

I would finally like to thank all my friends in Oxford (those who are still here and those who are gone). Meeting people from so many places in the world and with such different ideas has certainly made me more tolerant and open, probably just better.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

Sudden cardiac death following cardiac arrest is a major killer in industrialised societies. According to British Heart Foundation statistics[1], over 180,000 people died from cardiac arrest derived from cardiovascular disease (CVD) in the UK in 2009 (a third of all deaths). As well as human costs, treatment of CVD has major economic consequences: the estimated cost for the UK in 2006 was around £14.4 billion. The leading cause of sudden cardiac death is cardiac arrhythmias. Cardiac arrhythmias occur when the regular electrical activation of cardiac tissue that coordinates contraction becomes disturbed. The resulting uncoordinated sequence of contractions severely compromises the heart's mechanical pumping function. Among this family of pathologies, ventricular fibrillation is the most dangerous form of arrhythmia. Furthermore, electrical defibrillation through the application of strong electric shocks to the heart is the only effective therapy against ventricular fibrillation. This reality has turned cardiac electrophysiology and defibrillation into a key research topic in public health.

One of the main challenges in understanding cardiac arrhythmias and defibril-

[1] `www.heartstats.org`

lation is that the mechanisms involved span multiple spatial (at least from 10^{-9} to 10^{0}m, e.g. from protein complexes to a whole human being) and temporal scales (at least 10^{-7} to 10^{8}s, to capture behaviour from ion channel opening to disease development and ageing) [Rodríguez et al., 2010]. During the past decades, if not centuries, experimental techniques have provided useful insight into the main mechanisms underpinning cardiac electrophysiology. However, they also present inherent limitations in the spatial and temporal scales that can be directly observed and measured. Therefore, complementary methodologies such as multi-scale mathematical modelling and simulation have been applied to improve our understanding of cardiac electrophysiology and defibrillation.

In the past half century, modelling and simulation have been an essential tool in cardiac electrophysiology research. Several fields of science and engineering (e.g. physiology, medical imaging, mathematical modelling, numerical analysis, scientific computing) have met to enable simulations of cardiac electrophysiology using anatomically-based whole-organ models with detailed representation of membrane kinetics and myocardial fibre orientation. These simulations have helped provide insight into how the different spatiotemporal scales integrate to define normal and disturbed heart rhythms [Bishop et al., 2009; Kerckhoffs et al., 2006; Rodríguez et al., 2005; Vigmond et al., 2009]. The use of computational modelling and simulation also has a long history of productive insights and successful predictions relevant to mechanisms of arrhythmia, see [Trayanova et al., 2006] for a survey.

At a cellular level, over 40 models of cardiac electrophysiology of different species and cell types have been successfully developed and used for a variety of applications [Luo and Rudy, 1991; Mahajan et al., 2008; Maleckar et al., 2008; Noble, 1960; ten Tusscher and Panfilov, 2006; Zhang et al., 2000]. The models include representation of the main mechanisms of ionic transport across the cell membrane and between subcellular compartments. From a mathematical point of view, the models typically

consist of systems of ordinary differential equations (ODEs), with the most detailed having over 60 ODEs. These models allow representation of the effect of genetic mutations, drugs and disease on ion channel function.

At a tissue level, simulation of electrical propagation through cardiac tissue is generally performed by solving a system of partial differential equations, the bidomain model, or its simplification, the monodomain model. Simulation of defibrillation however requires representation of both intracellular and extracellular spaces, and therefore the bidomain model is required. Representation of anatomy and structure is also important in simulation of arrhythmia and defibrillation. Recent advances in imaging modalities have allowed a better characterisation of cardiac structure [Burton et al., 2006] and current ventricular models include highly-detailed representations of cardiac structures such as blood vessels, papillary muscles, the Purkinje network and fibre orientation. However, this comes with the cost of an increase in problem size and therefore computational burden. To date, bidomian models of cardiac defibrillation of small animals have been successfully developed [Bishop et al., 2009; Rodríguez et al., 2005]. Models of human cardiac electrophysiology have also been developed for a number of applications, e.g. study of conduction velocity and comparison of the monodomain and bidomain models [Potse et al., 2006], ventricular fibrillation [ten Tusscher et al., 2007], and cardiac resynchronisation therapy [Niederer et al., 2011b]. However, to the best of our knowledge, simulation studies of defibrillation mechanisms have only been performed using small animal ventricular geometries. The main limiting factor is the computational complexity associated with the solution of the bidomain equations with precise description of fibre structure on spatially very accurate human geometries made of tens of millions of grid points and with detailed representation of membrane kinetics.

In this scenario the development of simulation technology that can efficiently use the most advanced high performance computing (HPC) resources currently available

will be pivotal in enabling investigation of clinically relevant questions related to cardiac defibrillation mechanisms. As simulation technology becomes more complex new challenges arise related to how software is engineered and developed with aspects of correctness and software durability highlighted as critical [Linge et al., 2009; Pitt-Francis et al., 2008].

The remainder of this chapter has the following structure: Section 1.2 presents the scientific question we would like to tackle, namely the study of shock-induced arrhythmogenesis in the human heart, Section 1.3 describes the computational and numerical challenges faced, Section 1.4 introduces the software engineering perspective of the problem, Section 1.5 summarises the Thesis' goals, and Section 1.8 gives an outline of the rest of this document.

1.2 Application of interest: Shock-induced arrhythmogenesis in the human heart

Electrical defibrillation through the application of strong electric shocks to the heart is an effective therapy against ventricular fibrillation. In practice, this therapy is known not to be 100% effective. Computer simulation has played a pivotal role in improving our understanding of mechanisms of shock-failure and in particular of the shock-success dependency on shock-induced virtual electrode polarisation [Trayanova et al., 2006]. However, simulation studies of defibrillation have been performed with rabbit ventricular models [Ashihara et al., 2008; Maharaj et al., 2008; Rodríguez et al., 2005]. Authors often highlight that the extrapolation of their results to the human heart has to be made with caution due to the differences in size, anatomy, and electrophysiology between the rabbit and the human heart. To date, limitations on simulation technology have prevented the acquisition of results using human models. Limitations stem from the need of solving the bidomain equa-

tions with precise description of fibre structure on spatially very accurate human geometries and with detailed representation of membrane kinetics. Furthermore, a large number of simulations with high spatiotemporal resolution must be performed in order to explore parameter spaces and characterise the response to cardiac shocks.

In the next section, we identify the lack of freely available cardiac electrophysiology solvers at the level of performance and functionality necessary to run such simulations. We propose ways of bridging this gap and, therefore, of enabling the study of shock-induced arrhythmogenesis in the human heart. The development of efficient computational and numerical methods that take advantage of large-scale high performance computing (HPC) infrastructures is critical to enable computational studies of defibrillation mechanisms, as proposed here.

1.3 Computational and numerical challenges

A large body of literature concerns the development of cardiac electrophysiology solvers and the numerical methods employed (see [Bordas et al., 2009; Clayton et al., 2011; Linge et al., 2009] for surveys). Only a few of the solvers [Niederer et al., 2011b; Reumann et al., 2009; Vázquez et al., 2011] have been reported to run efficiently on large scale HPC resources. However, they share one or more of the following limitations relevant to the study of defibrillation: i) they use the monodomain model and therefore can not be used to simulate defibrillation, ii) they use an explicit time discretisation which is known to be conditionally stable, iii) they use spatial discretisation methods that only allow the use of regular grids (e.g. finite difference method), and/or iv) they are not freely available to the scientific community. In this Thesis, we aim at developing open source cardiac simulation technology that achieves a similar degree of performance in large scale HPC infrastructures to the aforementioned works using the bidomain equations, a finite element method space discretisation, and a semi-implicit time discretisation.

The finite element method (FEM) is preferred for the spatial discretisation of the bidomain equations for its support of unstructured grids. In brief, FEM recasts the original problem into the solution of a linear systems of equations: a system matrix and right-hand side (RHS) are assembled from the problem description and its temporal evolution. Explicit time discretisation imposes a constraint on the maximum timestep directly proportional to the grid edge length. In this scenario, the continuous increase in the level of detail of the anatomically-based models inevitably leads to increasingly shorter timesteps. This situation is likely to have a negative impact on overall performance. We will therefore consider unconditionally stable methods only (e.g. implicit or semi-implicit). Finally, we believe that open sourcing our simulation technology is: i) the only way to facilitate the reproducibility of our results, and ii) a contribution towards the shared scientific endeavour.

In addition to developing appropriate numerical schemes for the discretisation of the bidomain equations, improving the computational efficiency of the solver also requires the design of parallel algorithms for: i) mesh load and partitioning, ii) FEM system matrix assembly, iii) solution of cell-level electrophysiology, iv) FEM RHS assembly, and v) linear system solution. The two main challenges that we face are parallel scalability and algorithmic efficiency.

Parallel scalability is defined as the capability of a computer program to increase total throughput under an increased load when more computational units (e.g. CPU cores) are used for its execution. We will tackle two aspects of the problem: load balancing and communication reduction. Scalable implementations of the simulator components mentioned above are presented in this Thesis. Algorithmic efficiency describes how much of the various types of computer resources (e.g. CPU time, memory) an algorithm consumes. In later chapters, we will see that the solution of the linear system arising from the FEM discretisation of the bidomain equations is the simulation stage taking the largest proportion of total execution time. Therefore,

we will present novel numerical techniques (e.g. problem-specific preconditioners for Krylov subspace methods) in order to improve the efficiency of the linear system solution stage and by extension of the overall simulator.

1.4 Software development challenges

The complexity associated with cardiac electrophysiology simulation requires the use of dedicated software packages for the solution of the bidomain equations. The simulators most commonly used to date are CARP[2], CMISS[3], CONTINUITY[4], and MEMFEM (see [Niederer et al., 2011b] for a survey) as attested by publications such as [Bishop et al., 2009; Plank et al., 2008; Trayanova et al., 2002; Usyk and McCulloch, 2003]. However, each of them presents at least one of the following limitations (although not necessarily all of them):

1. Little stress has been put on using professional software engineering techniques (e.g. test-driven development). Reliability and correctness are therefore potentially compromised.

2. The degree of parallel performance is not adequate for our application of interest.

3. The source code is not freely available to the scientific community. This renders external evaluation of correctness difficult and reduces the chances of reproducing results.

In order to overcome the limitations listed before, the author (jointly with other researchers at the University of Oxford and external collaborators) has developed a new software package called Chaste. Chaste stands for "Cancer, Heart and Soft Tissue Environment" and provides an integrated modelling and simulation environment

[2]http://carp.meduni-graz.at
[3]http://www.cmiss.org
[4]http://www.continuity.ucsd.edu

for multi-scale Computational Biology problems such as cardiac electromechanics, discrete tissue modelling, and soft tissue modelling.

The technical complexity of the software and the number of lines of code to be written requires a change in the way software has been traditionally developed in academia. That is the reason behind the adoption of professional software engineering techniques in the earlier stages of the Chaste project. Agile methods [Fowler, 2000] are used with special emphasis on test-driven development. The former ensures that developers can cope with a set of potentially changing requirements (as is often common in academic environments). The latter guarantees high quality standards.

1.5 Thesis goals

The main goal of this Thesis is to develop open source HPC-enabled technology for cardiac electrophysiology simulation and use it to run the first simulation study on human shock-induced arrhythmogenesis. More specific targets are:

1. To develop novel numerical and computational techniques in order to bring Chaste's parallel performance to the level required by computationally demanding studies of cardiac defibrillation in the human heart. More precisely, we will investigate the use of better preconditioning techniques, implement parallelisation via domain decomposition, and address load balancing and communication issues arising from the use of state-of-the-art High Performance Computing (HPC) platforms.

2. To develop a model of human cardiac defibrillation including patient-specific geometry, a precise description of fibre structure, and detailed representation of membrane kinetics. The geometrical model chosen did not originally include fibre orientation information. Therefore, we will extend Chaste with the

functionality required to generate fibre orientation automatically.

3. To use Chaste to conduct the first simulation study of shock-induced arrythmogenesis in the human heart. This will gauge the success of the previous two tasks.

1.6 Thesis original contributions

The main contributions of this Thesis are:

1. The development of an efficient and scalable FEM solver for the numerical solution of the bidomain equations in computational domains made of millions of degrees of freedom.

2. The development of a novel hybrid Krylov subspace solver for linear systems with multiple, not simultaneously available, right-hand sides.

3. The development of sequential and parallel mesh-independent preconditioners for bidomain linear systems.

4. The development of a model of cardiac defibrillation of the human heart including accurate representation of fibre structure and membrane kinetics and its use to run the first simulation study of cardiac defibrillation in the human heart.

1.7 Publications generated

The following publications have been generated, entirely or partially, based on work presented in this Thesis:

Bernabeu, M. O., Bishop, M. J., Pitt-Francis, J., Gavaghan, D., Grau, V., and Rodriguez, B. (2008). High performance computer simulations for the study of

biological function in 3d heart models incorporating fibre orientation and realistic geometry at para-cellular resolution. In *Computers in Cardiology, 2008*, pages 721–724.

Bernabeu, M. O., Corrias, A., Pitt-Francis, J., Rodriguez, B., Bethwaite, B., Enticott, C., Garic, S., Peachey, T., Tan, J., Abramson, D., and Gavaghan, D. J. (2009). Grid computing simulations of ion channel block effects on the ECG using 3D anatomically-based models. In *Computers in Cardiology, 2009*, pages 213–216.

Bernabeu, M. O. and Kay, D. (2011). Scalable parallel preconditioners for an open source cardiac electrophysiology simulation package. *Procedia Computer Science*, 4:821–830. Proceedings of the International Conference on Computational Science, ICCS 2011.

Bernabeu, M. O., Pathmanathan, P., Pitt-Francis, J., and Kay, D. (2010a). Stimulus protocol determines the most computationally efficient preconditioner for the bidomain equations. *IEEE Transactions on Biomedical Engineering*, 57(12):2806 –2815.

Bernabeu, M. O., Southern, J., Wilson, N., and Pitt-Francis, J. (2011a). Chaste: a case study of parallelisation of an open source finite element solver with applications to computational cardiac electrophysiology simulation. *International Journal of High Performance Computing Applications*. Accepted subject to revision.

Bernabeu, M. O., Wallman, M., and Rodriguez, B. (2010b). Shock-induced arrhythmogenesis in the human heart: A computational modelling study. In *Engineering in Medicine and Biology Society (EMBC), 2010 Annual International Conference of the IEEE*, pages 760–763.

Bernabeu, M. O., Wallman, M., and Rodriguez, B. (2011b). Shock-induced arrhyth-

mogenesis in a human model with patient-specific anatomy. In *Heart Rhythm, 32nd Annual Scientific Sessions, 2011.*

Pathmanathan, P., Bernabeu, M. O., Bordas, R., Cooper, J., Garny, A., Pitt-Francis, J. M., Whiteley, J. P., and Gavaghan, D. J. (2010). A numerical guide to the solution of the bidomain equations of cardiac electrophysiology. *Progress in Biophysics and Molecular Biology*, 102(2-3):136–155.

1.8 Thesis outline

This Thesis has the following structure. Chapter 2 provides an introduction to computational cardiac electrophysiology. Chapter 3 describes the numerical methods employed for the solution of the bidomain equations. Chapter 4 describes the efficient parallel implementation of a complete bidomain solver in modern HPC infrastructures, including novel results in load balancing and communication reduction in the underlying finite element method solver. Chapter 5 presents the problem of efficient preconditioning for a range of bidomain linear systems and describes two novel techniques that achieve substantial reduction of solution time for certain problem configurations. Chapter 6 describes the effective parallelisation of the aforementioned preconditioners and identifies a dependency of the most efficient bidomain preconditioner on the problem characteristics. Chapter 7 firstly describes the development of a human model of cardiac defibrillation with patient-specific anatomy including a generic framework for fibre orientation generation and automatic transmural heterogeneities definition and secondly presents the results of the first simulation study of shock-induced arrhythmogenesis in the human heart. Finally, Chapter 8 brings together the conclusions of the investigations presented in the previous chapters and proposes future lines of research.

Chapter 2

Computational modelling and simulation of cardiac defibrillation

In this Chapter, we provide an introduction to cardiac electrophysiology and to the main modelling and simulation tools used for the investigation of cardiac defibrillation processes. Section 2.1 begins by describing cardiac anatomy and function, followed by an introduction to the biophysical processes underlying electrical activation of the myocardium (at a cell, tissue, and organ level). The last part of the section describes disturbances in the normal electrical activation, known as cardiac arrhythmias, and how they are treated by means of electrical shocks. Section 2.2 gives a brief introduction to the mathematical models used to study cardiac electrophysiology at cell and tissue/whole-organ level with particular emphasis on how they have been applied to the study of the mechanisms of cardiac defibrillation.

Some of the material in this chapter is based on [Bishop, 2008; Clayton et al., 2011; Henriquez, 1993; Trayanova et al., 2006].

2.1 The electrophysiology of the heart

The heart is a muscular organ found in all animals with a circulatory system (including all vertebrates), that is responsible for pumping blood throughout the blood vessels by repeated, rhythmic contractions. In this section, we will first introduce the anatomy and function of the heart and describe the basic mechanisms of cardiac electrical activation and how they integrate to sustain the mechanical function. In the second part, we will describe a family of pathologies related with abnormal activity in the heart known as cardiac arrhythmias and one of their potential treatments: electrical cardiac defibrillation.

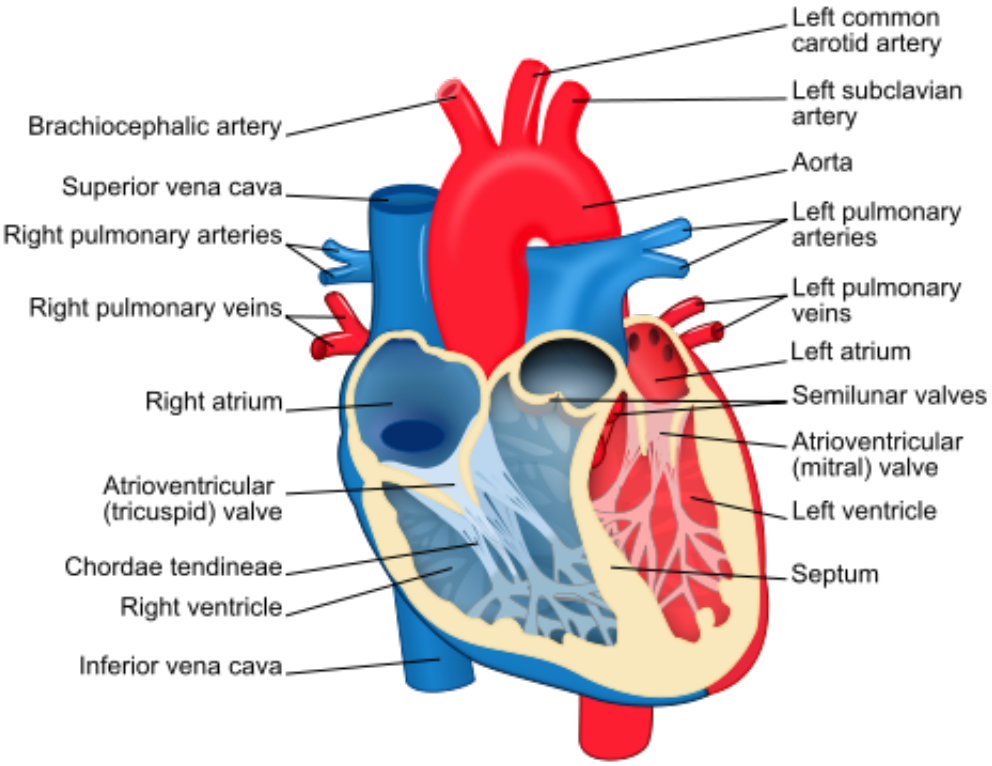
2.1.1 Cardiac anatomy and function

The fundamental anatomy of the heart is shown in Figure 2.1. The human heart has a mass of between 250 and 350 grams and is about the size of a fist. It is located anterior to the vertebral column and posterior to the sternum. It is enclosed in a double-walled sac called the pericardium. This sac protects the heart, anchors its surrounding structures, and prevents overfilling of the heart with blood.

The human heart has four chambers, two superior atria and two inferior ventricles. The atria are the receiving chambers and the ventricles are the discharging chambers. As can be seen in Figure 2.1, the myocardial walls of the left ventricle (LV) are significantly thicker than those of the right ventricle (RV), as the LV is required to pump blood around the entire body, whereas the RV must only pump blood to nearby lungs and back.

The pathway of blood through the human heart consists of a pulmonary circuit and a systemic circuit. Deoxygenated blood flows through the heart in one direction, entering through the superior vena cava into the right atrium (RA) and is pumped through the tricuspid valve into the right ventricle before being pumped out through the pulmonary valve to the pulmonary arteries into the lungs. It returns from the

Figure 2.1 Structure diagram of the human heart from an anterior view. Blue components indicate de-oxygenated blood pathways and red components indicate oxygenated pathways. Image originally released under Creative Commons Attribution and ShareAlike license.

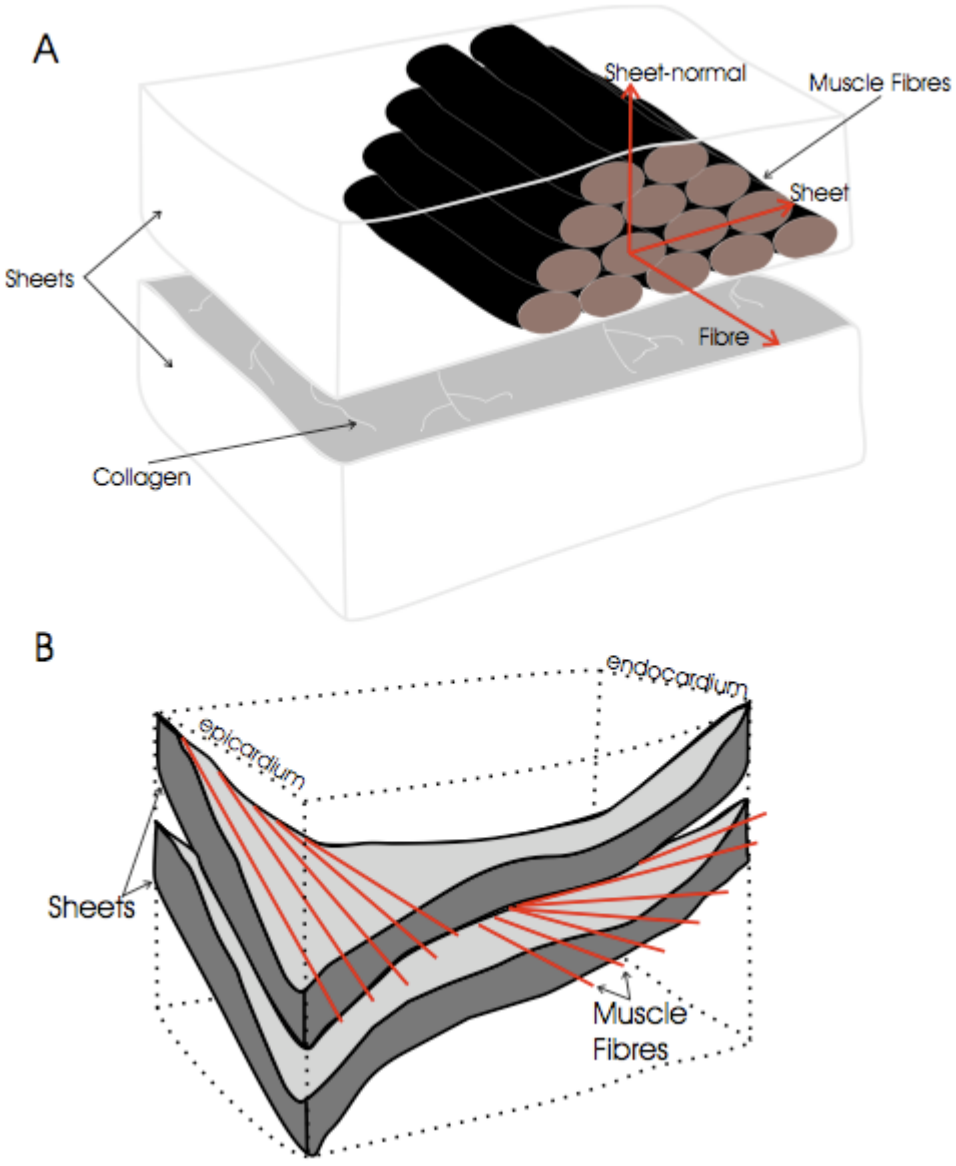


lungs through the pulmonary veins to the left atrium where it is pumped through the mitral valve into the left ventricle before leaving through the aortic valve to the aorta.

The outer wall of the human heart is composed of three layers. The outer layer is called the epicardium. The middle layer is called the myocardium, which makes up the bulk of the heart wall. The inner layer is called the endocardium and is in contact with the blood that the heart pumps. Also, it merges with the inner lining of the blood vessels and covers the heart valves. The tissue separating the two ventricles is known as the septum. The apex is the blunt point situated in an inferior (pointing down and left) direction. The base of the heart, directed superiorly, posteriorly, and to the right is formed mainly by the left atrium, and, to a small extent, by the back part of the right atrium.

The structural architecture of the ventricular myocardial wall is highly heterogeneous, and is composed of inter-connected sheets of tissue, typically 3 to 4 cells in thickness, separated by cleavage planes. Cardiac cells, or myocytes, lie longitudinally within the plane of the sheets to form cardiac fibres which are bound together by collagen [Legrice et al., 1997]. The basic sheet structure of the myocardium is shown in Figure 2.2. For modelling purposes, the arrangement of cardiac fibres provides a useful natural set of mutually-orthogonal material directions aligned with the important structural features of the myocardial tissue. The microstructurally-defined coordinate system varies at different positions throughout the heart, and deforms along the tissue. The three respective axes chosen define: i) the fibre direction, along the myocytes; ii) the sheet direction, perpendicular to the fibre within the plane of the sheet; and, iii) the sheet-normal direction, perpendicular to both the fibre and the sheet. Electrical activation propagates preferentially along the fibre direction and to a lesser extent along the sheet direction, the least preferential direction of propagation is normal to the sheet.

Figure 2.2 Myocardial structure: a) Schematic diagram of the microstructure of the myocardial wall, showing the arrangement of cardiac fibres within layered sheets of tissue as well as the microstructurally-defined cardiac coordinate system, shown by the red arrows, b) Representation of the change in fibre orientation between epicardium and endocardium, through the myocardial wall. Image originally from [Bishop, 2008], adapted from [LeGrice et al., 1995].

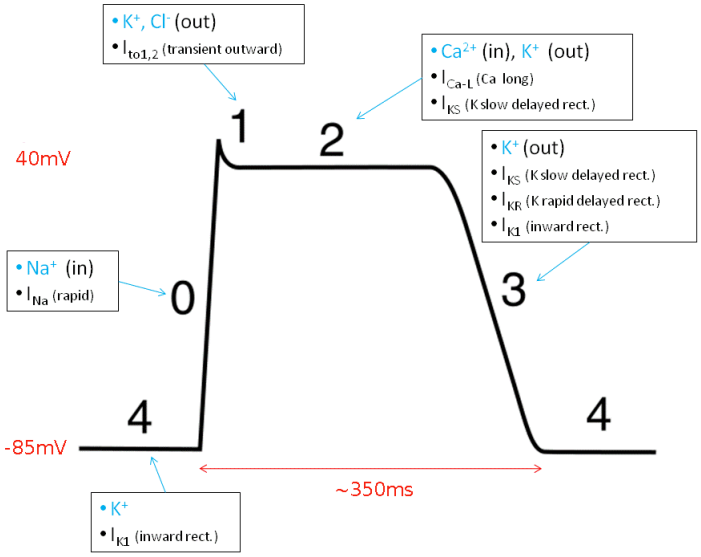


In the next subsections, we present the basic mechanisms governing electrical activation at a cell (i.e. cardiac action potential) and a tissue level.

2.1.2 The cardiac action potential

A cardiac myocyte is an electrically active system. A potential difference exists across the cell membrane, called the transmembrane potential, V_m. Along the cell membrane a variable number of ion channels, pumps and exchangers can be found. Throughout the cardiac cycle, the transmembrane potential of the cell experiences changes due to ions flowing into and out of the cell in a sequence of well-defined stages. The change in the electrical state of the cell over time is called the cardiac action potential (AP). The action potential has 5 phases. Figure 2.3 depicts them.

Figure 2.3 Idealised cardiac AP along with the main currents and ionic species involved. Modified from an image originally released under Creative Commons Attribution and ShareAlike license.



Phase 0

Phase 0 is the rapid depolarization phase. The largest slope of phase 0 represents the maximum rate of depolarization of the cell. This phase is due to the opening

of the fast Na^+ channels causing a rapid increase in the membrane conductance to Na^+ (G_{Na}) and thus a rapid influx of Na^+ ions (I_{Na}) into the cell (i.e. the fast Na^+ current).

Phase 1

Phase 1 of the action potential occurs with the inactivation of the fast Na^+ channels. The transient net outward current causing the small downward deflection of the action potential is due to the movement of K^+ and Cl^- ions, carried by the I_{to1} and I_{to2} currents, respectively. Particularly the I_{to1} contributes to the "notch" of some ventricular myocyte action potentials.

Phase 2

This "plateau" phase of the cardiac action potential is sustained by a balance between inward movement of Ca^{2+} (I_{Ca}) through L-type calcium channels and outward movement of K^+ (I_{Ks}). The sodium-calcium exchanger current, $I_{Na,Ca}$ and the sodium/potassium pump current, $I_{Na,K}$ also play minor roles during phase 2.

Phase 3

During phase 3 (the "rapid repolarization" phase) of the action potential, the L-type Ca^{2+} channels close, while the slow delayed rectifier (I_{Ks}) K^+ channels are still open. This ensures a net outward current, corresponding to negative change in membrane potential, thus allowing more types of K^+ channels to open. These are primarily the rapid delayed rectifier K^+ channels (I_{Kr}) and the inwardly rectifying K^+ current, I_{K1}. This net outward, positive current (equal to loss of positive charge from the cell) causes the cell to repolarise. The delayed rectifier K^+ channels close when the membrane potential is restored to about -80 to -85 mV, while I_{K1} remains conducting throughout phase 4, contributing to set the resting membrane potential.

Phase 4

Phase 4 is the resting membrane potential. This is the period that the cell remains in until it is stimulated by an external electrical stimulus (typically an adjacent cell). This phase of the action potential is associated with diastole of the chamber of the heart. In addition to stimulus from adjacent cells, certain cells of the heart have the ability to undergo spontaneous depolarization, in which an action potential is generated without any influence from nearby cells. This is known as cardiac muscle automaticity.

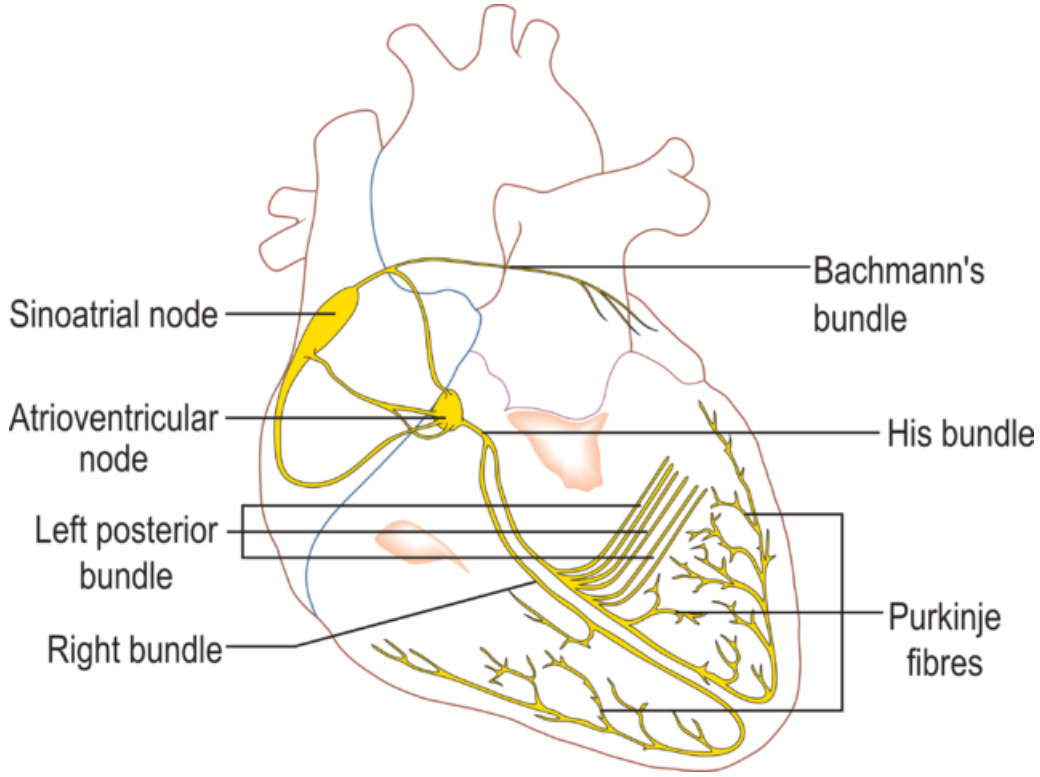
2.1.3 Tissue propagation

The heart contracts as a result of electrical stimulation. Under physiological conditions, stimulation propagates across the myocardium in the form of a single smooth wavefront causing individual cells to depolarise and contract. This contraction happens in a very coordinated and synchronised way, efficiently pumping blood around the body.

The initial stimulus begins in a small group of specialised pacemaker cells in the sinoatrial (SA) node, at the top of the RA. Because cardiac muscle cells are electrically coupled, impulses from the SA node spread rapidly through the walls of the atria, causing both atria to contract in unison. The impulses also pass to another region of specialized cardiac muscle tissue, a relay point called the atrioventricular (AV) node, located in the wall between the right atrium and the right ventricle. Here, the impulses are delayed for about 0.1s before spreading to the walls of the ventricle. The delay ensures that the atria empty completely before the ventricles contract. Except for the AV node, the tissue at the junction of the atria and the ventricles is non-conducting to prevent the direct spread of excitation between the atria and the ventricles. From the AV node, the electrical propagation enters the bundle of His which carries the stimulus down the septum to each ventricle, via the

right and left bundle branches which are electrically isolated from the myocardium. At the apex of the heart, each branch is subdivided into a complex network of fibres called the Purkinje system, which delivers the stimulus to the endocardial surface of the ventricles. As the electrical activation enters the myocardial wall of the ventricles, it spreads upwards and outwards across the tissue, causing cardiac cells to contract. Figure 2.4 shows a diagram of this dedicated conduction system.

Figure 2.4 Isolated conduction system of the heart with its main components highlighted. Image originally released under Creative Commons Attribution and Share-Alike license.



2.1.4 Cardiac arrhythmia

The term arrhythmia covers a wide range of medical conditions involving abnormal electrical activity in the heart and leading to sinus rhythm perturbation. More specifically, during arrhythmic episodes the heart beat may become faster, slower or more irregular than in normal conditions.

Several types of arrhythmia are asymptomatic and not associated with increased mortality. Others, such as those inducing palpitations, are noticeable by the patient but harmless. However, arrhythmic episodes such as fibrillation are potentially life-threatening [Jalife, 2000]. Fibrillation is often characterised by the appearance of high frequency circulating reentrant waves (i.e. reentries) that mask the normal rhythmic electrical signal originating from the SA node. This asynchronous electrical activation leads to out-of-phase localised contractions. If reentrant episodes become self-sustained they can impede the normal pumping function of the heart to an extent that the body's demand of oxygen is not fully fulfilled.

Arrhythmias can affect both the atria and the ventricles. Atrial arrhythmias are not directly life-threatening, but are known to lead into other complications such as increased risk of stroke [Wolf et al., 1991]. The most fatal forms of arrhythmias are generally those affecting the ventricles, as the loss of pumping efficiency deprives oxygen from all other organs in the body and can lead to cardiac arrest and sudden cardiac death if not treated promptly.

The most potentially lethal type of ventricular arrhythmia is ventricular fibrillation (VF), which is very often preceded by ventricular tachycardia (VT) [Pogwizd and Corr, 1990]. VT is characterised by reentrant spiral-wave patterns of electrical activation across the myocardium. As a result, the excitation of the heart remains reasonably periodic, and large regions of the ventricles still contract at the same time, in a fairly coordinated fashion. Pumping efficiency is therefore reduced although not enough to induce cardiac arrest. VF follows VT when the spiral-wave patterns of excitation associated with VT break-up into multiple fractionated wavefronts of activity in a process called spiral-wave break-up [Panfilov, 1998]. The resulting completely unsynchronised and uncoordinated contraction patterns lead to an almost complete loss of cardiac output.

Causes of cardiac arrhythmias can vary but they often include drug action, dis-

ease or genetic mutations. The majority of the reentrant arrhythmias are initiated by the interaction of a propagating wavefront with an obstacle in its path, generating reentrant circuits [Jalife, 2000]. This obstacle may appear due to either structural or functional heterogeneity in the properties of the myocardium. Functional changes to the electrophysiological properties of myocardium generally result from some form of cardiovascular-related disease that is the trigger for the vast majority of the arrhythmias [Pogwizd and Corr, 1990]. The most common of these is coronary heart disease, whereby the blood vessels supplying the heart with oxygen get clogged with fatty deposits. As a result, the tissue becomes ischaemic. If the lack of blood supply persists, damage and potential death of the tissue can ensue, this is known as myocardial infarction. Such significant changes in the affected area of myocardium can result in corresponding large changes in tissue conductivity, or in the AP duration of the affected cells [Elharrar and Zipes, 1977]. Consequently, the tissue here may take longer to recover following the passage of an initial electrical wavefront. Thus, if a second wavefront rapidly follows the first, conduction is blocked in the diseased tissue, as the cells here are still refractory[1]. Heterogeneous dispersion of repolarisation can cause the electrical wavefronts to rotate or potentially fractionate around the refractory tissue, triggering arrhythmias.

2.1.5 Cardiac defibrillation

Electrical defibrillation through the application of strong electric shocks to the heart is an effective therapy against lethal arrhythmias such as VF. Clinically, these shocks are delivered either externally, via electrical paddles placed on the patient torso, or directly to the heart, via implantable devices called internal cardioverter defibrillators.

Although widely adopted, this treatment is unfortunately not always success-

[1]The refractory period of a cardiac myocyte comprises phases 0–3 in Figure 2.3. Once an AP has started, no new AP can be triggered until transmembrane potential returns to resting potential

ful. Despite years of research, the mechanisms behind defibrillation failure remain incompletely understood. In order for a shock to be successful it must extinguish existing pre-shock reentrant wavefronts, and, importantly, it must not initiate new fibrillatory episodes following the shock [Trayanova, 2001]. The latter point is relevant because a large body of research has demonstrated that the application of a strong electrical shock to the heart can in fact induce ventricular arrhythmias (i.e. shock-induced arrhythmogenesis). It is well accepted nowadays that an electrical shock can induce VF when applied within a certain period of time (known as vulnerable window or VW) following the beginning of a cardiac cycle. Furthermore, [Chen et al., 1986] found that arrhythmias can only be induced by shock strengths between a minimum and a maximum value, known in the literature as the lower and upper limits of vulnerability (LLV and ULV). A shock applied during this period of time and for this range of shock strengths is said to be within the vulnerability area (VA). Experimentally, a strong correlation between the ULV and the defibrillation threshold (the shock-strength required for arrhythmia termination) has been shown to exist [Chen et al., 1986]. This has lead to the widely-held view that the mechanisms by which defibrillation shocks re-initiate VF are strongly linked to the mechanisms by which a shock induces VF if applied within the VA following pacing. Therefore, in order to facilitate a better understanding of defibrillation and arrhythmogenesis by failed shocks, cardiac vulnerability to electric shocks has also been investigated [Rodríguez et al., 2005; Trayanova et al., 2006; Trayanova and Tice, 2009] using both experimental and theoretical techniques. In the next section, we present the mathematical models used for theoretical investigation of defibrillation mechanisms.

2.2 Multi-scale models of cardiac electrophysiology

In this section we introduce the mathematical models used for the study of cardiac electrophysiology at a cell and a tissue level with particular emphasis on how they have been applied for the study of arrhythmia and cardiac electrical defibrillation.

2.2.1 Action potential models

Following the seminal work of Hodgkin and Huxley, Denis Noble published the first model of the cardiac Purkinje AP in 1960 [Noble, 1960]. This has been recognised as a breakthrough in the field of cardiac electrophysiology, since it set the mathematical background for the study of biological processes that had been experimentally observed for years. Since then, a large number of mathematical models of the cardiac AP have been developed: not only for Purkinje cells but for a wide range of cell types (e.g. ventricular myocytes [DiFrancesco and Noble, 1985; Luo and Rudy, 1991; Mahajan et al., 2008; ten Tusscher and Panfilov, 2006], atrial myocytes [Maleckar et al., 2008; Nygren et al., 1998], sinoatrial cells [Demir et al., 1994; Zhang et al., 2000]) over a wide range of species (e.g. mice, rat, rabbit, guinea pig, human).

AP mathematical models describe the evolution of the different cellular ionic concentrations and currents in the cell by means of ordinary differential equations (ODEs) (or stochastic differential equations, depending on the modelling requirements). In its most abstract form, this is a system of the form

$$\frac{\partial \mathbf{u}}{\partial t} = \mathbf{f}(\mathbf{u}, V)\,, \tag{2.1}$$

$$I_{\text{ion}} = g(\mathbf{u}, V)\,, \tag{2.2}$$

where V is the transmembrane potential of the cell, I_{ion} is the current flowing through

the cell membrane, $\mathbf{u}$ is a vector containing cell-level variables, such as ionic concentrations and membrane gating variables, and the functions $\mathbf{f}$ and g are determined by the characterisation of ionic channels, pumps and exchangers used in a particular model.

The complexity of these systems of ODEs varies but the most detailed ones include over 60 variables. One of the advantages of this modelling approach is that the numerical solution of ODEs is a problem well understood. Therefore, a wide range of algorithms and software tools exist for their accurate and efficient solution [Cooper et al., 2006; Sundnes et al., 2009].

The CellML project [Cuellar et al., 2003] has carried out an important task of standardisation in the field. CellML is a XML-based markup language for the description of mathematical models. Although it could, in principle, characterise any mathematical model, CellML is primarily used to describe models relevant to the field of biology. A comprehensive collection of published models can be found on the CellML website.

Figure 2.5 shows a schematic diagram of the ten Tusscher and Panfilov 2006 model [ten Tusscher and Panfilov, 2006]. The outer-most rectangle represents the membrane of a human myocyte. Several ionic channels and exchangers allow current to flow inwards and/or outwards as defined by the direction of the arrows. All the channels use a Hodgkin and Huxley formulation except the calcium-induced calcium release channel which is modelled as a four-state Markov model. The area inside the rectangle represents the intracellular space, which is subsequently divided into the sub-membrane space, sarcoplasmic reticulum and the bulk of the cytoplasm. Note that extra channels are defined across their boundaries.

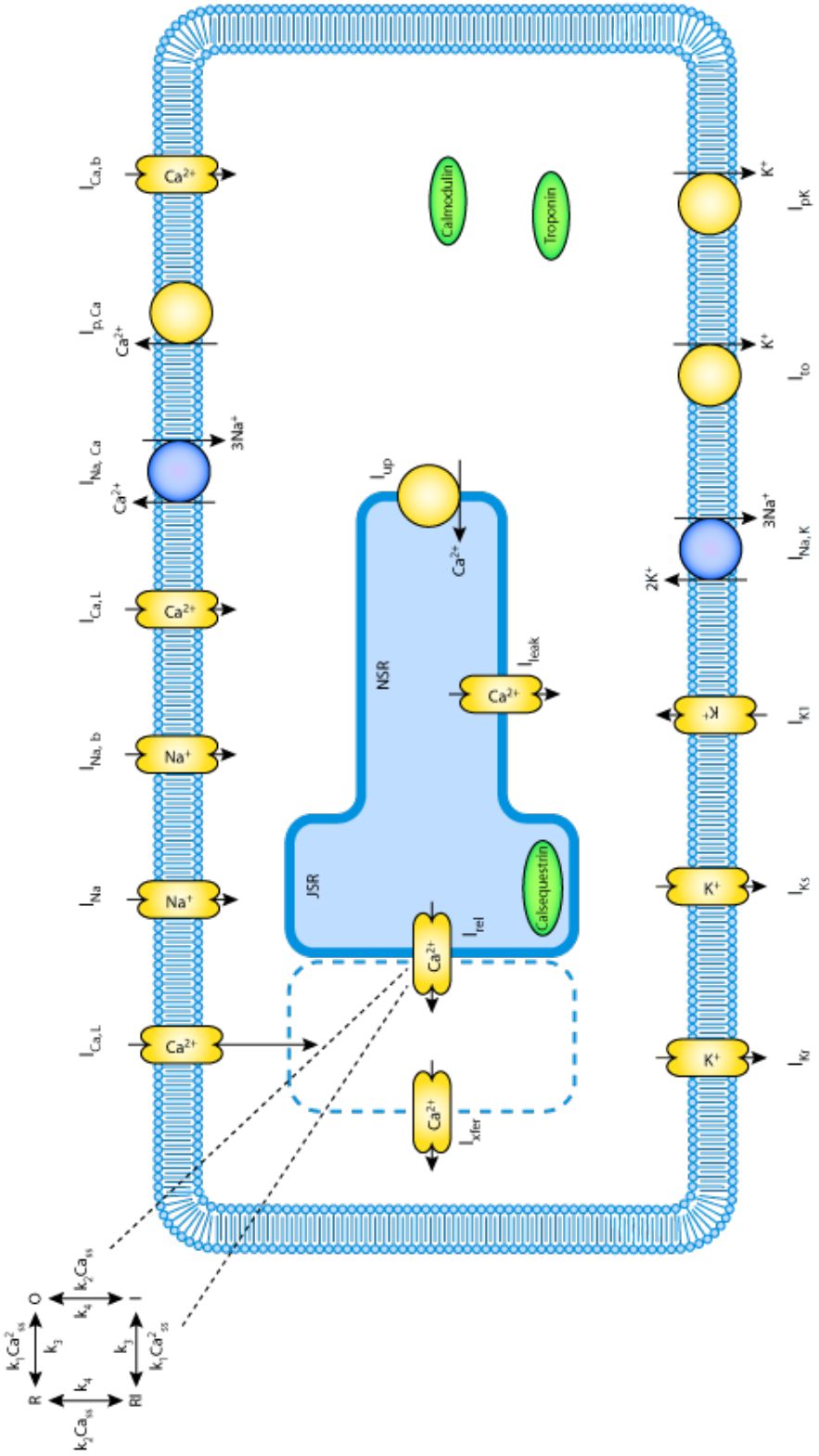


Figure 2.5 A schematic diagram describing the ion movement across the cell surface membrane and the sarcoplasmic reticulum, which are described by the ten Tusscher and Panfilov 2006 mathematical model of the human ventricular myocyte. Image from http://www.cellml.org.

2.2.2 The bidomain equations

The complex microstructure of cardiac muscle comprised of coupled cells, enveloped by an interstitium[2] made up of blood vessels, connective tissue, and fluid, presents some obvious problems for the study of tissue as an electrical medium. The bidomain model overcomes this difficulty by considering tissue not as a discrete structure, but rather as two coupled, continuous (i.e. homogenised) domains: one for the intracellular space and the other for the interstitial/extracellular space [Keener and Sneyd, 1998, Chapter 11] [Henriquez, 1993; Tung, 1978]. Therefore, the averaged potentials and currents in both domains are defined at every point in space. The tissue microstructure is partially preserved by assigning a conductivity tensor at each point (according to the system of coordinates described in Section 2.1.1). One advantage of this homogenised approach is that the governing equations of the electric fields become partial differential equations that can be solved numerically (we will describe how in Chapter 3).

The bidomain model relies on the cable theory, which has been experimentally shown to be a suitable way of modelling mammalian cardiac fibres [Clerc, 1976; Weidmann, 1970]. [Schmitt, 1969] was the first to propose the bidomain concept, but it was not until a decade later that it was described in mathematical terms by [Tung, 1978]. Section 2.2.3 presents a historical account on the use of the bidomain model for the study of cardiac defibrillation. We will now present a derivation of the bidomain equations.

Bidomain model derivation

Let Ω be the tissue region of interest and define two overlapping computational domains: the intracellular domain, Ω_i, and the extracellular domain, Ω_e, separated by the cell membrane. Let ϕ_i and ϕ_e be the potential in the intracellular and

[2]The term interstitium refers to the space between cells in a tissue.

extracellular domains, respectively. The description of each domain is based on a generalised version of Ohm's law defining the relationship between the electric field vector $\mathbf{E}$ (in Vm^{-1}) derived from the potential ϕ (in V), the current density vector $\mathbf{J}$ (in Am^{-2}), and the conductivity tensor σ (in Sm^{-1}):

$$\mathbf{E} = -\boldsymbol{\nabla}\phi\,, \tag{2.3}$$

$$\mathbf{J} = \sigma\mathbf{E} = -\sigma\boldsymbol{\nabla}\phi\,. \tag{2.4}$$

Considering the intracellular and extracellular spaces separately, we have:

$$\mathbf{J}_i = -\sigma_i\boldsymbol{\nabla}\phi_i \qquad \text{in } \Omega_i\,, \tag{2.5}$$

$$\mathbf{J}_e = -\sigma_e\boldsymbol{\nabla}\phi_e \qquad \text{in } \Omega_e\,, \tag{2.6}$$

where $\mathbf{J}_{i,e}$ and $\sigma_{i,e}$ are the intracellular and extracellular current densities and conductivity tensors, respectively.

Considering the conservation of current and charge, and assuming only membrane related sources in the intracellular and extracellular spaces we can write divergence equations:

$$\nabla\cdot\mathbf{J}_i = -I_m \qquad \text{in } \Omega_i\,, \tag{2.7}$$

$$\nabla\cdot\mathbf{J}_e = I_m \qquad \text{in } \Omega_e\,, \tag{2.8}$$

where I_m (in Am^{-3}) is the transmembrane current per unit volume. I_m is composed of a capacitive component I_c (in Am^{-2}), resulting from the semi-permeable nature of the cell membrane, and an ionic component I_{ion} (in Am^{-2}), resulting from the current flow through ion channels, pumps and exchangers in the cell membrane (as

described by one of the ionic models presented in Section 2.2.1):

$$I_m = \chi(I_c + I_{\text{ion}}) = \chi(\mathcal{C}\frac{\partial V}{\partial t} + I_{\text{ion}})\,, \tag{2.9}$$

where χ (in m^{-1}) is the surface area-to-volume ratio of a cardiac cell, $\mathcal{C}$ (in Fm^{-2}) the cell membrane capacitance, and V the transmembrane potential:

$$V = \phi_i - \phi_e\,. \tag{2.10}$$

Combining (2.5)–(2.9) and adding two terms $I_i^{(\text{vol})}$ and $I_e^{(\text{vol})}$ representing intracellular and extracellular stimuli per unit volume (such that $I_i^{(\text{vol})} + I_e^{(\text{vol})} = 0$), we obtain the bidomain equations:

$$\nabla \cdot \sigma_i(\boldsymbol{\nabla}\phi_i) = \chi\left(\mathcal{C}\frac{\partial V}{\partial t} + I_{\text{ion}}\right) + I_i^{(\text{vol})} \qquad \text{in } \Omega_i\,, \tag{2.11}$$

$$\nabla \cdot \sigma_e(\boldsymbol{\nabla}\phi_e) = -\chi\left(\mathcal{C}\frac{\partial V}{\partial t} + I_{\text{ion}}\right) + I_e^{(\text{vol})} \qquad \text{in } \Omega_e\,. \tag{2.12}$$

Boundary conditions for the bidomain model usually assume that there is no current across the boundary that enters directly into the intracellular space, whereas if there is an injected current, it enters the tissue through the extracellular domain [Keener and Sneyd, 1998, Chapter 11]

$$\mathbf{n} \cdot (\sigma_i \boldsymbol{\nabla}\phi_i) = 0 \qquad \text{on } \partial\Omega_i\,, \tag{2.13}$$

$$\mathbf{n} \cdot (\sigma_e \boldsymbol{\nabla}\phi_e) = I_e^{(\text{surf})} \qquad \text{on } \partial\Omega_e\,, \tag{2.14}$$

where $I_e^{(\text{surf})}$ is the extracellular current per unit area applied across the boundary and $\mathbf{n}$ is the outward-facing unit normal.

This is the most basic form of the bidomain equations and it does not account for the tissue being contained in an electrically passive surrounding medium (as

required by defibrillation studies). We will consider that extension in Chapter 3.

The bidomain model provides an accurate description of electrical propagation within cardiac tissue suitable for a number of applications. The bidomain model has been extensively used for the study of defibrillation processes, see [Trayanova et al., 2002, 2006] for surveys. In the next subsection we will summarise some of the findings.

2.2.3 Importance of the bidomain model in the study of defibrillation

The development and adoption of the bidomain model is considered a breakthrough in the field of cardiac electrophysiology research. The bidomain model has played a fundamental role in the development of new theories of the processes underpinning cardiac defibrillation beyond what experimental techniques have traditionally allowed. More interestingly, hypotheses developed with the bidomain model have been confirmed experimentally *a posteriori*, once the appropriate technology has become available [Trayanova et al., 2006]. In this section we will survey some of those advances.

Bidomain simulations presented in [Sepulveda et al., 1989] demonstrated that tissue response to a strong unipolar stimulus involved simultaneous occurrence of positive (depolarising) and negative (hyperpolarising) effects in close proximity. This process is referred in the literature as virtual electrode polarisation (VEP) and was in disagreement with previous belief of response being unipolar. Subsequent optical mapping studies confirmed the prediction [Wikswo et al., 1995].

Further studies [Roth, 1995] demonstrated that the close proximity of a de-excited region and a virtual cathode could result in excitation at shock end (known in the literature as "break excitation"). The virtual cathode serves as an electrical stimulus eliciting a propagating wave in the newly created excitable area. Positive

virtual electrode polarisation can also result in depolarisation in regions where tissue is at or near resting potential (known in the literature as "make excitation"). Make excitation happens at onset of stimulus as opposed to the break excitation that happens at shock-end. This characterisation provided the necessary framework for understanding how a defibrillation shock results in the development of new postshock activations.

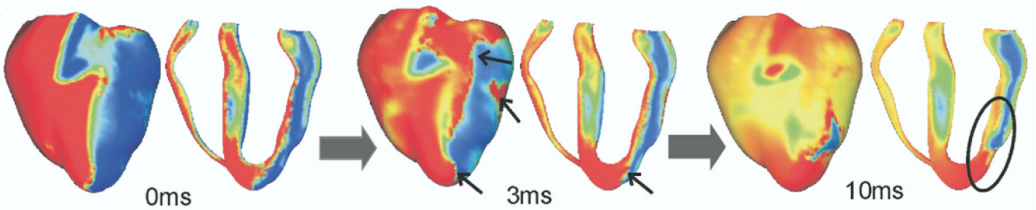
The next big contribution of bidomain modelling was the detailed analysis of virtual electrode polarisation dependence on cardiac tissue structure and the configuration of the applied field. Theoretical considerations led to the recognition of two types of VEP: i) surface VEP, which penetrates the ventricular wall over a few cell layers; and ii) bulk VEP throughout the ventricular wall [Trayanova et al., 1998]. Analysis of the bidomain equations revealed that necessary conditions for the existence of the bulk VEP are: i) unequal anisotropy ratios for the intracellular and the extracellular domains of the myocardium and ii) either spatial nonuniformity in applied electric field, nonuniformity in tissue architecture (i.e. changes in fibre orientation) or structural discontinuity (i.e. cleavage planes and blood vessels) [Hooks et al., 2002; Trew et al., 2009].

Bidomain simulations in two dimensions [Skouibine et al., 2000] demonstrated that the outcome of a defibrillation shock depends crucially on the propagation of newly formed postshock excitations through shock-induced excitable areas. The shock succeeds in extinguishing fibrillatory wavefronts and not initiating new reentry if the excitations manage to traverse the excitable areas before the rest of the myocardium recovers from shock-induced depolarisation. The onset of a break excitation is associated with formation of a phase singularity: a point along the boundary between shock-induced depolarisation and de-excitation [Efimov et al., 1998]. If these phase singularities survive, thus allowing development of a spiralling wavefront, the shock results in (re)initiation of reentrant arrhythmia. The establishment

of a reentrant arrhythmia in the ventricles thus strongly depends on the anatomic features and geometric asymmetries of the cardiac chamber [Rogers, 2002]. This finding highlighted the importance of considering realistic 3D geometries (or much more elaborate lower dimensional models) in order to unveil the mechanisms behind shock-failure.

Further inquiry into the 3D mechanisms governing the failure of defibrillation has been hampered by the inability of current experimental techniques to resolve electrical behaviour confined to the depth of the ventricles during and after the shock. During the early 2000's, simulation technology advances allowed the development of bidomain models with 3D ventricular geometries [Rodríguez et al., 2005]. These studies confirmed that the understanding acquired in earlier, simpler bidomain studies remains valid in 3D ventricular geometries. Furthermore, these simulations revealed the importance of accounting for the geometry of the ventricular chambers in the quest to understand generation of postshock arrhythmias. For example, the difference in the thickness of the ventricular walls is ultimately manifested as a preferential location of the postshock excitable area. These simulations showed that the postshock excitable areas were always located in the thick left ventricle and septum, but never in the thin right ventricle (Figure 2.6).

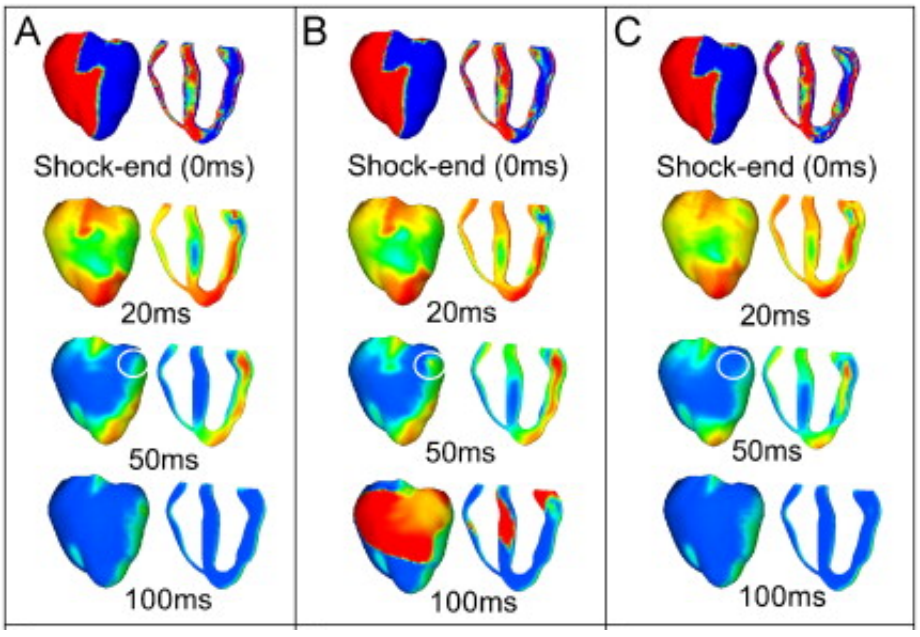
Figure 2.6 Postshock transmembrane potentials in the rabbit ventricles. Virtual electrode polarisation at shock-end and activity at 3 and 10 ms postshock, demonstrating break excitations (small arrows) and propagation through the postshock excitable areas (encircled region). Image originally from [Trayanova et al., 2006].



Recent work [Maharaj et al., 2008] has shown that inclusion of transmural heterogeneity in ionic currents results in an increase in vulnerability to shocks, in particular

by increasing ULV and enlarging VW. These changes in vulnerability are related to an increase in transmural dispersion in repolarisation and, therefore, to the likelihood of establishment of re-entrant circuits. In contrast, reduced sub-epicardial coupling results in decrease in both ULV and VW. This decrease is caused by altered VEP around the region of sub-epicardial uncoupling, and specifically, by the increase in: i) the amount of positively polarised myocardium at shock-end and ii) the spatial extent of post-shock wavefronts ([Maharaj et al., 2008]). Figure 2.7 compares the three scenarios.

Figure 2.7 Transmembrane potential at shock-end (0 ms panel) and at 20, 50, and 100 ms following a 30.5 V/cm shock applied at a coupling interval of 140 ms to the homogeneous ventricles (panel A), to the heterogeneous ventricles (panel B), and to heterogeneous ventricles with sub-epicardial uncoupling (panel C). Image originally from [Maharaj et al., 2008].



In [Ashihara et al., 2008], the authors proposed a new theory of shock-induced arrhythmogenesis following strong shocks near the upper limit of vulnerability. In this scenario, formation of virtual electrode polarisation, quick re-excitation, and synchronous repolarisation across the ventricles took place sequentially. However, a wavefront that originated deep within the wall remained submerged, giving rise to an

isoelectric window, until it broke through onto the epicardium and then propagated, resulting in intramural reentry.

The Purkinje system has also been implicated in the failure of defibrillation shocks. Studies with rabbit ventricular models incorporating free-running His-Purkinje systems [Vigmond and Clements, 2007] demonstrated that the Purkinje fibres might be involved in maintaining VF in the early postshock period by providing alternative propagation pathways. The presence of Purkinje fibres also led to re-entry initiation at lower shock strengths [Vigmond et al., 2009].

All the 3D studies cited so far were performed with rabbit ventricular geometries because: i) the reduced size of the rabbit heart decreases the computational burden of the simulations, and ii) lack of availability of human cardiac geometries at the appropriate resolution. Furthermore, authors of studies with rabbit geometries [Maharaj et al., 2008; Rodríguez et al., 2004] often highlight that the extrapolation of their results to the human heart has to be made with caution due to the differences in size and anatomy between a human and a rabbit heart. In the following section, we describe two of the rabbit ventricular geometries most widely used.

2.3 Anatomically-based models of the heart

As highlighted in the previous section, most studies of defibrillation have been conducted using ventricular models of small mammals and specifically rabbit. In this section we present two rabbit ventricular geometrical models, namely *UCSD* and *Oxford* rabbit models, used in studies of defibrillation. Both represent the geometry of both ventricles of a rabbit heart with an unstructured tetrahedral mesh and are suitable for the solution of the bidomain equations with the finite element method. Both models have been used in this Thesis for the development of numerical and computational techniques presented in Chapter 4–6 as well as in a large number of other publications [Ashihara et al., 2008; Bernabeu et al., 2008, 2010a; Bishop and

Plank, 2011; Corrias et al., 2010; Rodríguez et al., 2005; Rodríguez and Trayanova, 2003; Trayanova et al., 2002]. For their reduced size they are a good testbed before moving into the large scale human simulations presented in Chapter 7.

2.3.1 The UCSD rabbit model

The UCSD rabbit model is based upon anatomical data obtained from a study of a single rabbit heart by [Vetter and McCulloch, 1998]. The original anatomical dataset provided information regarding both the structural geometry of the ventricles and the orientation of cardiac fibres throughout the tissue. The original mesh, which contained just 36 linear cubic-Hermite hexahedral elements and 99 nodes, was re-meshed to generate a volumetric tetrahedral finite element grid, using the method described in [Trayanova et al., 2002]. In this Thesis, we will use a version that does not include representation of the blood in the ventricles or the surrounding media. Within the myocardium, an internal node spacing of 500 μm was used. The mesh consists of 63,885 nodes and 322,269 elements. Figure 2.8 shows an image of the mesh.

Figure 2.8 UCSD rabbit model

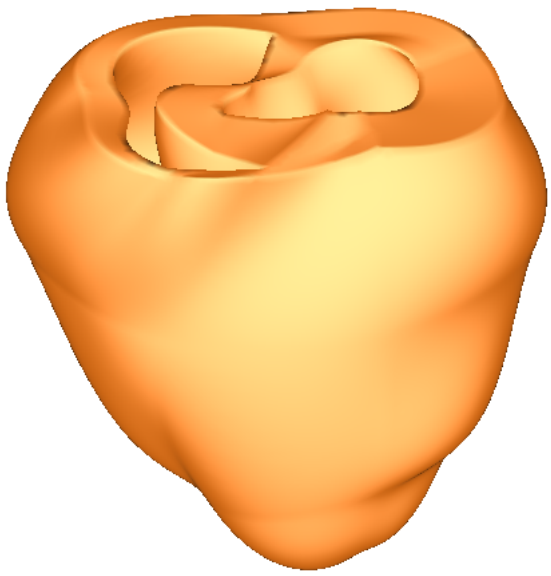
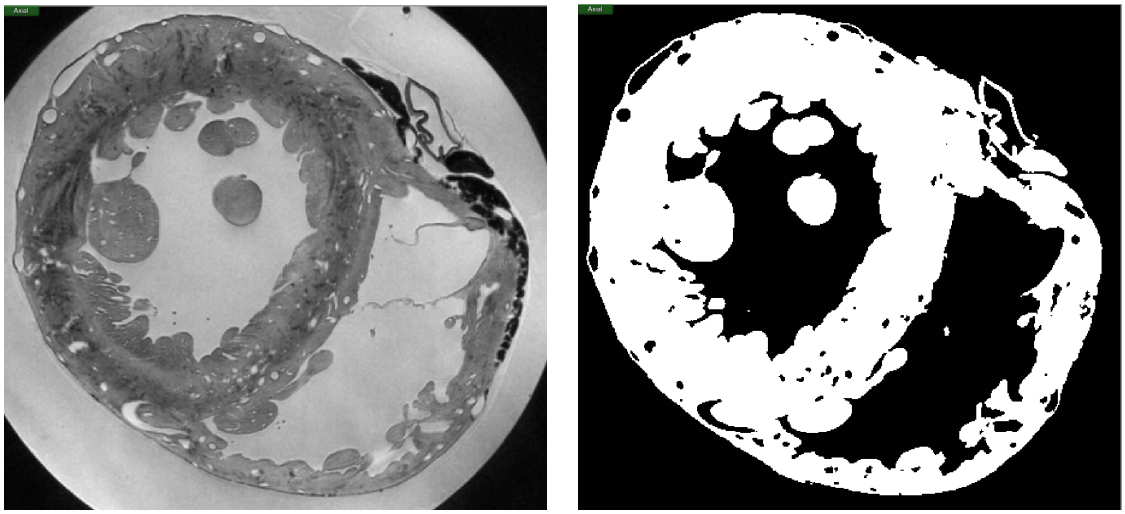


Figure 2.9 Anatomical high-resolution MR image (a) with final result of segmentation pipeline (b). Images originally from [Bernabeu et al., 2008]



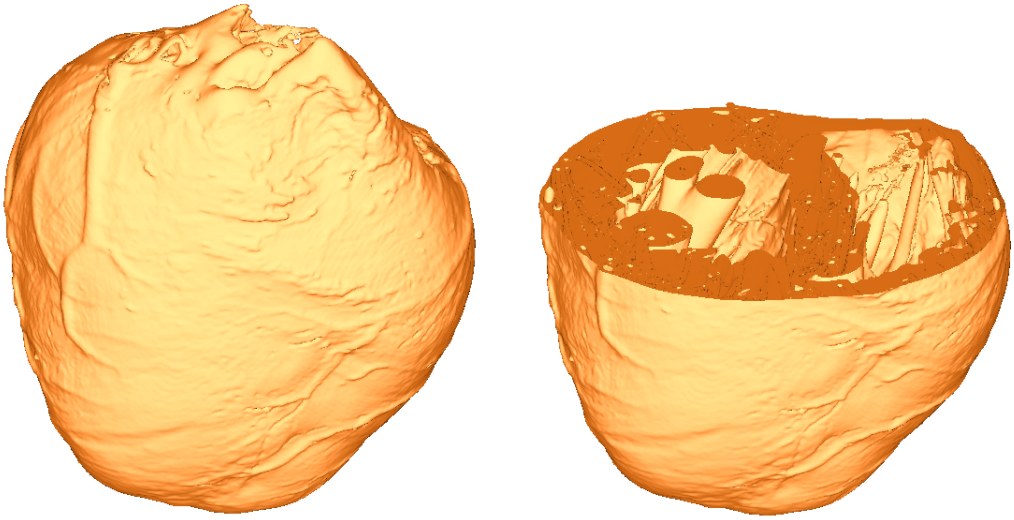
(a) (b)

2.3.2 The Oxford rabbit model

The Oxford rabbit model [Bishop et al., 2009] is the rabbit tetrahedral mesh with the highest level of detail currently available, with an average edge length of 130 μm. The model is based upon magnetic resonance (MR) images. MR scans were performed on a fixed and agar embedded female rabbit heart using an 11.7 T (500 MHz) MR system. For specific details of the MR system and scan protocols used see [Burton et al., 2006]. Scans were acquired with an in-plane resolution of 26.4 μm $\times$ 26.4 μm and out-of-plane resolution of 24.4 μm producing a MR image stack containing $1024 \times 1024 \times 2048$ voxels. Due to computational memory constraints, however, this was downsampled once by a factor of 2 using Matlab (The MathWorks, Inc.) to produce a more managable data set containing $512 \times 512 \times 1024$ voxels at the expense of losing some fine-grained detail. Figure 2.9a shows a cross-section of the MR data set taken along the short-axis of the heart. The downsampled data set was segmented and post-processed using a segmentation pipeline application based on level sets algorithms and developed using the Insight Toolkit libraries[3]. The final

[3] `www.itk.org`

Figure 2.10 Finite element rabbit ventricular mesh. Images originally from [Bernabeu et al., 2008]



(a) (b)

result of the segmentation can be seen in Figure 2.9b. The final segmented and post-processed data set was used to generate a tetrahedral finite element mesh using the meshing software Tarantula (`www.meshing.at`). Tarantula is an Octree-based mesh generator that generates unstructured, boundary fitted, locally refined conformal finite element meshes, and has been custom designed for cardiac modelling.

The final mesh consisted of $3,717,056$ finite element node points making-up $20,401,874$ tetraheral elements (with mean edge-length of approximately 130 μm) and with $1,194,070$ bounding triangular faces. Epicardial as well as left ventricular (LV) and right ventricular (RV) endocardial surfaces were defined as separate surfaces. Figure 2.10 shows the final cardiac mesh; panel (a) shows the full data set, whilst (b) shows a slice along the short axis.

In Chapters 5 and 6, versions of the original mesh remeshed at $h = 250$ μm and $h = 480$ μm will also be used. For the extracellular shock experiments the three meshes were embedded into a cube-shaped control volume representing the bath surrounding the heart. We will refer to these geometries as the high, low, and

very low resolution rabbit mesh, with and without bath, respectively. Table 2.1 summarises their characteristics.

Table 2.1 Details of realistic 3D meshes used.

Resolution	Bath	Edge length	# of nodes	# of elements
Low	No	250 μm	572,066	3,172,910
High	No	125 μm	4,310,704	24,217,344
Very low	Yes	480 μm	195,815	1,137,153
Low	Yes	250 μm	1,091,855	6,444,768
High	Yes	125 μm	6,449,098	38,299,065

Chapter 3

Computational and numerical methods for cardiac electrophysiology problems

The previous chapter provided an introduction to the theory of electrical cardiac defibrillation and presented a derivation of the bidomain model. The bidomain model of cardiac electrophysiology is a coupled system of ordinary and partial differential equations (ODEs and PDEs) that describe the evolution of intracellular and extracellular potential in cardiac tissue. In order to simulate the bidomain model on computers we need to perform a finite discretisation of the original continuous model. Several spatial discretisation methods have been successfully applied to the bidomain equations: the finite difference method [Potse et al., 2006; Pullan et al., 2005], the finite volume method [Coudiere and Pierre, 2006; Jacquemet and Henriquez, 2005], and the finite element method (FEM) [Pathmanathan et al., 2010; Vigmond et al., 2002] along with appropriate temporal discretisations: explicit [Penland et al., 2002; Pollard and Barr, 1991; Vigmond et al., 2002], fully-implicit [Munteanu and Pavarino, 2009; Murillo and Cai, 2004] and semi-implicit [Colli-Franzone and Pavarino, 2004; Whiteley, 2006].

The choice of spatiotemporal discretisation method is driven by the requirements of our application: i) use of large anatomically-detailed geometries, ii) need for efficient parallel implementations, and iii) numerical stability and low numerical error for the problem configurations of interest. Based on this criteria, we have chosen:

1. A semi-implicit time discretisation where the reactive part (i.e. cell-level electrophysiology) and the diffusive part of the equations are decoupled and treated explicitly and implicitly, respectively. The choice of a semi-implicit scheme is a tradeoff between the stability of a fully implicit method and the low computational cost of an explicit method.

2. FEM spatial discretisation for its support of unstructured grids. This characteristic is very desirable since it allows easy representation of complex irregular geometries. In brief, FEM recasts the original problem into the solution of a linear systems of equations; system matrix and right-hand side (RHS) are assembled from the problem description and its temporal evolution.

The rest of the chapter has the following structure: Section 3.1 gives a brief introduction to the finite element method, Section 3.2 presents the solvers and preconditioners used for the solution of the linear system arising from the finite element discretisation of PDEs, Section 3.3 presents the semi-implicit FEM solution of the bidomain equations used in this work, finally Section 3.4 introduces the software platform where all the solutions presented in this Thesis have been implemented: Chaste.

Some of the material in Sections 3.1–3.3 is based on [Barrett et al., 1994; Elman et al., 2005; Gutknecht and Röllin, 2002; Johnson, 1987; Osborne, 2009; Pathmanathan et al., 2010; Pitt-Francis et al., 2009; van der Vorst, 2003].

3.1 An introduction to the finite element method

A large proportion of the mathematical models in science and engineering take the form of differential equations. Only in the simplest cases, or under strong assumptions, it is possible to find exact analytical solutions to the equations in the model. Often, one has to rely on numerical techniques for finding approximate solutions for particular parameter sets using computers. The finite element method is a general technique for the numerical solution of differential and integral equations. The method was introduced by engineers in the late 1950's and early 1960's for the numerical solution of PDEs in structural engineering. When the mathematical study of the finite element method started in the mid 1960's it soon became clear that in fact the method is a general technique for numerical solution of partial differential equations with roots in the variational methods in mathematics introduced at the beginning of the 20^{th} century.

3.1.1 Model problem

In order to illustrate the fundamental building blocks required for the solution of PDEs with the finite element method, we will use the following parabolic PDE model problem in two dimensions (known in the literature as the heat equation): (P1) Let Ω be a bounded open domain with boundary $\partial\Omega = \partial\Omega_1 \cup \partial\Omega_2$ and $\mathbf{I} = (0, T]$ a finite period of time, find u defined in $\Omega \times \mathbf{I}$ such that

$$\frac{\partial u}{\partial t} - \Delta u = f \qquad \text{in} \quad \Omega \times \mathbf{I}\,, \tag{3.1}$$

$$u = 0 \qquad \text{on} \quad \partial\Omega_1\,, \tag{3.2}$$

$$\frac{\partial u}{\partial \mathbf{n}} = g \qquad \text{on} \quad \partial\Omega_2\,, \tag{3.3}$$

$$u(\mathbf{x}, 0) = u_0(\mathbf{x}) \qquad \mathbf{x} \in \Omega \tag{3.4}$$

where the 2D Laplacian operator, Δ, is defined by

$$\Delta u := \frac{\partial^2 u}{\partial x_1^2} + \frac{\partial^2 u}{\partial x_2^2}\,, \tag{3.5}$$

f is a function defined in Ω, g is a function defined on $\partial\Omega_2$, and u_0 is the initial condition for equation (3.1). Equations (3.2) and (3.3) are Dirichlet and Neumann boundary conditions, respectively. Let $\mathbf{n}(\mathbf{x})$ be the outward pointing normal vector to $\partial\Omega$ at $\mathbf{x} \in \partial\Omega$ and

$$\boldsymbol{\nabla} u := \begin{pmatrix} \frac{\partial u}{\partial x_1} \\ \frac{\partial u}{\partial x_2} \end{pmatrix}\,, \tag{3.6}$$

then the normal derivative is defined by

$$\frac{\partial u}{\partial \mathbf{n}} := \boldsymbol{\nabla} u \cdot \mathbf{n}\,. \tag{3.7}$$

3.1.2 Weak form and function spaces

We will now show that the solution u of the boundary problem (P1) (i.e. the strong solution) is also the solution of an equivalent variational/weak problem. To formulate it we introduce the space of square integrable functions on Ω

$$L_2(\Omega) := \{v : v \text{ is defined on } \Omega \text{ and } \int_\Omega v^2\,\mathrm{d}x < \infty\}\,, \tag{3.8}$$

and the more restrictive spaces

$$H^1(\Omega) := \{v \in L_2(\Omega) : \int_\Omega |\boldsymbol{\nabla} v|^2\,\mathrm{d}x < \infty\}\,, \tag{3.9}$$

and

$$H_0^1(\Omega) := \{v \in H^1(\Omega)\ : v = 0 \text{ on } \partial\Omega_1\}\,. \tag{3.10}$$

With respect to these spaces we define the following inner products

$$(v, w) := \int_{\Omega} vw \, \mathrm{d}x \qquad \forall v, w \in L^2(\Omega)\,, \tag{3.11}$$

$$a(v, w) := \int_{\Omega} \boldsymbol{\nabla} v \boldsymbol{\nabla} w \, \mathrm{d}x \qquad \forall v, w \in H^1(\Omega)\,, \tag{3.12}$$

$$\langle v, w \rangle := \int_{\partial\Omega_2} vw \, \mathrm{d}S \qquad \forall v, w \in L^2(\Omega)\,. \tag{3.13}$$

In order to derive the weak form of (P1), we multiply (3.1) by a test function $v \in V = H_0^1(\Omega)$ and integrate over the domain to obtain

$$\left(\frac{\partial u}{\partial t}, v\right) - (\Delta u, v) = (f, v)\,. \tag{3.14}$$

Applying the divergence or Green-Gauss' theorem[1]

$$-\int_{\Omega} \Delta u v \, \mathrm{d}x = \int_{\Omega} \boldsymbol{\nabla} u \boldsymbol{\nabla} v \, \mathrm{d}x - \int_{\partial\Omega} \frac{\partial u}{\partial \mathbf{n}} v \, \mathrm{d}s\,, \tag{3.15}$$

using (3.3) and the fact that $v = 0$ on $\partial\Omega_1$, we reduce the system to one containing at most first-order derivatives

$$\left(\frac{\partial u}{\partial t}, v\right) + a(u, v) = (f, v) + \langle g, v \rangle\,. \tag{3.16}$$

Hence, the variational or weak formulation of (P1) is: (P2) Find $u(t) \in V,\ t \in \mathbf{I}$ such that

$$\left(\frac{\partial u(t)}{\partial t}, v\right) + a(u(t), v) = (f, v) + \langle g, v \rangle \qquad \forall v \in V\,. \tag{3.17}$$

[1] See [Johnson, 1987] for a derivation of the equality.

3.1.3 Temporal discretisation

The first step towards the finite element solution of (P2) is to discretise u in time. Let $I_k := (t_{k-1}, t_k],\ k = 1, \ldots, n_t$, be a non-overlapping partition of $\mathbf{I}$ such that

$$\mathbf{I} = \bigcup_{k=1}^{n_t} I_k \tag{3.18}$$

with $\Delta t := t_k - t_{k-1}$ constant for simplicity. Let $u^{\Delta t} \in V$ be a piecewise constant in time function

$$u^{\Delta t}(x,t) := u(x,t_k) : t \in I_k \qquad \forall x \in \Omega \tag{3.19}$$

and define $u^k := u^{\Delta t}(\cdot, t_k)$ (and similarly with f^k and g^k). The time derivative $\frac{\partial u}{\partial t}$ can be approximated with the quotient $\frac{u^{k+1}-u^k}{\Delta t}$ with discretisation error $O(\Delta t)$ such that the semi-discrete (in time) approximation of (P2) is

$$(u^{k+1}, v) + a(u^{k+\theta}, v)\Delta t = (u^k, v) + (f^{k+\theta}, v)\Delta t + \langle g^{k+\theta}, v\rangle \Delta t \qquad \forall v \in V\,, \tag{3.20}$$

where, for $0 \leq \theta \leq 1$,

$$u^{k+\theta} = \theta u^{k+1} + (1-\theta)u^k\,. \tag{3.21}$$

This is known as the theta method.

It can be shown that for $\theta \in [\frac{1}{2}, 1]$ this method is unconditionally stable, however for $\theta \in [0, \frac{1}{2})$ the scheme is conditionally stable. In one spatial dimension and $\theta \in [0, \frac{1}{2})$ the method is stable for

$$\Delta t \leq \frac{h^2}{6(1-2\theta)}(1-\varepsilon), \quad 0 < \varepsilon < 1, \tag{3.22}$$

[Süli, 2007], where ε is a fixed number used in the derivation of the bound and h is a parameter depending on the spatial discretisation that we will introduce in the following subsection. When $\theta = 0$, (P2) reduces to the method known as explicit or

forward Euler: Find $u^k \in V,\ k = 1, \ldots, n_t$ such that

$$(u^{k+1}, v) = -a(u^k, v)\Delta t + (u^k, v) + (f, v)\Delta t + \langle g, v\rangle \Delta t \qquad \forall v \in V\,. \tag{3.23}$$

When $\theta = 1$, (3.20) reduces to the implicit or backward Euler method: (P3) Find $u^k \in V,\ k = 1, \ldots, n_t$ such that

$$(u^{k+1}, v) + a(u^{k+1}, v)\Delta t = (u^k, v) + (f, v)\Delta t + \langle g, v\rangle \Delta t \qquad \forall v \in V\,. \tag{3.24}$$

Due to stability issues in (3.23) we will choose (3.24) for the rest of the discussion.

3.1.4 Finite element spatial discretisation

Although (P3) is discrete in time, it is still a continuous (infinite dimensional) system in space, not suitable for being solved with computers. In the current section, we will derive the spatially discrete finite element weak form of (P3). Let us start by constructing a finite-dimensional subspace V_h of V. For simplicity we will take Ω to be a polygonal 2D domain. A tessellation of Ω is made by subdividing Ω into a set $T_h = \{T_1, T_2, \ldots, T_m\}$ of non-overlaping open triangles T_i, with vertices $N_i, i = 1, \ldots, n$, such that

$$\Omega = \bigcup_{i=1}^{m} T_i\,. \tag{3.25}$$

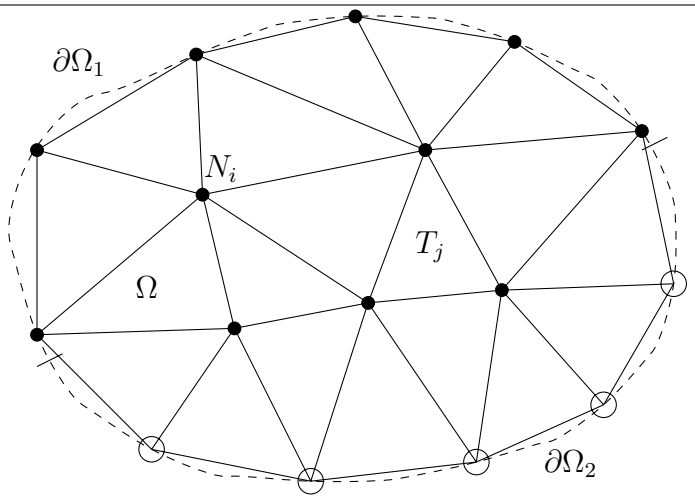
We assume, for simplicity, that no vertex of one polygon lies on the edge of any another polygon. Figure 3.1 shows an example of domain discretisation. We define the mesh parameter h to be the longest edge in $T_i,\ \forall T_i \in T_h$.

We define the finite element space, V_h, as follows:

$$V_h = \{v \in C(\bar{\Omega}) \subset H_0^1(\Omega) : v|_T \text{ is linear for } T \in T_h\} \tag{3.26}$$

where $C(\bar{\Omega})$ is the space of continuous functions on $\bar{\Omega}$ and is known to be a subspace

Figure 3.1 Example of FEM tessellation of a 2D domain, Ω, with triangular elements. Circles along the boundary are nodes in $\partial\Omega_2$. Solid dots along the boundary are nodes in $\partial\Omega_1$. Adapted from [Osborne, 2009].



of $H_0^1(\Omega)$, and $v|_T$ denotes the restriction of v to T. Therefore, the space $V_h \subset V$ consists of all the functions in V that are linear on each triangle T, continuous, and by definition vanish on $\partial\Omega_1$.

We can now formulate the finite element weak form of (P3): (P4) Find $u_h^k \in V_h, k = 1, \ldots, N$ such that

$$(u_h^{k+1}, v) + a(u_h^{k+1}, v)\Delta t = (u_h^k, v) + (f, v)\Delta t + \langle g, v\rangle \Delta t \qquad \forall v \in V_h\,. \tag{3.27}$$

This method is usually referred to as Galerkin's Finite Element method.

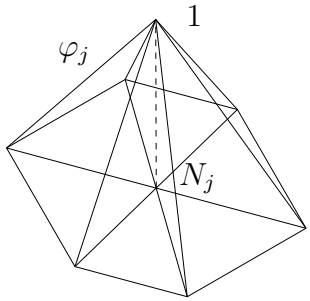
Since V_h is finite dimensional, we may construct a basis that spans it. The corresponding basis functions $\varphi_j \in V_h,\ j = 1, \ldots, n$ are then defined by

$$\varphi_j(N_i) := \delta_{ij} = \begin{cases} 1 & \text{if } i = j\,, \\ 0 & \text{otherwise}\,. \end{cases} \tag{3.28}$$

As parameters to describe a function $v \in V_h$ we choose the values of v at the nodes $N_i,\ i = 1, \ldots, n$, excluding the nodes on $\partial\Omega_1$ since $v = 0$ on $\partial\Omega_1$. Figure 3.2

shows that the support of φ_j (i.e. $\mathbf{x}$ such that $\varphi_j(\mathbf{x}) \neq 0$) consists of the polygons containing the common node N_j. It will be later seen that keeping the support of the

Figure 3.2 Support of basis function φ_j. $\varphi_j(N_j) = 1$. $\varphi_j(N_{i\neq j}) = 0$. Adapted from [Osborne, 2009].



basis functions low is important to increase the sparsity of the linear system arising from the FEM discretisation of a PDE and therefore reduce the computational burden associated with its solution. A function $v \in V_h$ may be represented

$$v(x) = \sum_{j=1}^{n} \eta_j \varphi_j(x), \quad \eta_j = v(N_j) \qquad \forall x \in \bar{\Omega}. \tag{3.29}$$

It can be observed that if $u_h \in V_h$ satisfies (3.27) then in particular

$$(u_h^{k+1}, \varphi_j) + a(u_h^{k+1}, \varphi_j)\Delta t = (u_h^k, \varphi_j) + (f, \varphi_j)\Delta t + \langle g, \varphi_j \rangle \Delta t \qquad \forall j = 1, \ldots, n\,, \tag{3.30}$$

and if these equations hold, then by taking linear combinations, it can be seen that u_h^k satisfies (3.27). Since

$$u_h^k(x) = \sum_{i=1}^{n} \xi_i^k \varphi_i(x), \quad \xi_i^k = u_h^k(N_i) \tag{3.31}$$

we can write (3.30) as

$$\sum_{i=1}^{n} \xi_i^{k+1}(\varphi_i, \varphi_j) + \sum_{i=1}^{n} \xi_i^{k+1} a(\varphi_i, \varphi_j)\Delta t = \sum_{i=1}^{n} \xi_i^k (\varphi_i, \varphi_j) + (f, \varphi_j)\Delta t + \langle g, \varphi_j \rangle \Delta t \qquad j = 1, \ldots, n\,, \tag{3.32}$$

which is a linear system of equations with n equations in n unknowns $\xi_1, \ldots, \xi_n$. In matrix form the linear system (3.32) can be written

$$(M + \Delta t K)\xi^{k+1} = M\xi^k + \Delta t \mathbf{c} + \Delta t \mathbf{d} \tag{3.33}$$

$$A\mathbf{x}^{k+1} = \mathbf{b}^k \tag{3.34}$$

where $M = [m_{ij}]$ and $K = [k_{ij}]$ are the $n \times n$ matrices with entries $m_{ij} = (\varphi_i, \varphi_j)$ and $k_{ij} = a(\varphi_i, \varphi_j)$ and where $\xi = (\xi_1, \ldots, \xi_n)^T$, $\mathbf{c} = (c_1, \ldots, c_n)^T$, and $\mathbf{d} = (d_1, \ldots, d_n)^T$ are n-vectors with $c_i = (f, \varphi_i)$ and $d_i = \langle g, \varphi_i \rangle$. Matrices M and K are often referred as the mass and stiffness matrices, with terminology from early applications of FEM in structural mechanics.

3.1.5 FEM linear system properties

Matrices M and K (and therefore A) are symmetric and positive definite since $(\varphi_i, \varphi_j) = (\varphi_j, \varphi_i)$, $a(\varphi_i, \varphi_j) = a(\varphi_j, \varphi_i)$ and with (3.29), we have

$$\sum_{i,j=1}^{n} \eta_i (\varphi_i, \varphi_j) \eta_j = (\sum_{i=1}^{n} \eta_i \varphi_i, \sum_{j=1}^{n} \eta_j \varphi_j) = (v, v) \geq 0\,, \tag{3.35}$$

$$\sum_{i,j=1}^{n} \eta_i a(\varphi_i, \varphi_j) \eta_j = a(\sum_{i=1}^{n} \eta_i \varphi_i, \sum_{j=1}^{n} \eta_j \varphi_j) = a(v, v) \geq 0\,, \tag{3.36}$$

with equality to zero in both equations if and only if $\boldsymbol{\nabla} v = 0$ (that is, since v is 0 on $\partial\Omega_1$, only if $v = 0$) or $\eta_j = 0$ for $j = 1, \ldots, n$. Recall that a symmetric $n \times n$ matrix $S = [s_{ij}]$ is said to be positive definite if

$$\eta^T S \eta = \sum_{i,j=1}^{n} \eta_i s_{ij} \eta_j > 0 \qquad \forall \eta \in \mathbb{R}^n, \eta \neq 0\,. \tag{3.37}$$

Also recall that a symmetric matrix S is positive definite if and only if the eigenvalues of S are strictly positive.

Since a positive definite matrix is non-singular it follows that the linear system (3.33) has a unique solution. It can also be observed that $A = [a_{ij}]$ is sparse, i.e. only a few entries a_{ij} are different from zero, and diagonally dominant. This very important property depends on the fact that a basis function φ_j of V_h is different from zero only on a few intervals and thus will interfere only with a few other basis functions, in our example $a_{ij} = 0$ unless N_i and N_j are nodes of the same triangle. Therefore, the structure of the matrix is known *a priori*. The fact that the basis functions may be chosen in this way is an important distinctive feature of the finite element method, allowing efficient computer implementations.

3.1.6 FEM linear system assembly

The entries of $A = [a_{ij}]$ are usually in practice computed by summing the contributions from the different triangles:

$$a_{ij} = a(\varphi_i, \varphi_j) + (\varphi_i, \varphi_j)\,, \tag{3.38}$$

$$a(\varphi_i, \varphi_j) = \sum_{T \in T_h} a(\varphi_i, \varphi_j)_T\,, \tag{3.39}$$

$$(\varphi_i, \varphi_j) = \sum_{T \in T_h} (\varphi_i, \varphi_j)_T\,, \tag{3.40}$$

where $a(\cdot,\cdot)_T, (\cdot,\cdot)_T$ are versions of the inner products described above restricted to a subdomain $T \in T_h$:

$$(v, w)_T := \int_T vw \,\mathrm{d}x \qquad \forall v, w \in L^2(\Omega)\,, \tag{3.41}$$

$$a(v, w)_T := \int_T \boldsymbol{\nabla} v \boldsymbol{\nabla} w \,\mathrm{d}x \qquad \forall v, w \in H^1(\Omega)\,. \tag{3.42}$$

It can be noticed that $a(\varphi_i, \varphi_j)_T = (\varphi_i, \varphi_j)_T = 0$ unless both nodes N_i and N_j are vertices of T.

Let N_i, N_j, and N_k be the vertices of the triangle $T \in T_h$. The 3×3 symmetric

element stiffness matrix for element T is

$$K_T = \begin{bmatrix} a(\varphi_i, \varphi_i)_T & a(\varphi_i, \varphi_j)_T & a(\varphi_i, \varphi_k)_T \\ & a(\varphi_j, \varphi_j)_T & a(\varphi_j, \varphi_k)_T \\ & & a(\varphi_k, \varphi_k)_T \end{bmatrix}, \tag{3.43}$$

and similarly with the element mass matrix.

$$M_T = \begin{bmatrix} (\varphi_i, \varphi_i)_T & (\varphi_i, \varphi_j)_T & (\varphi_i, \varphi_k)_T \\ & (\varphi_j, \varphi_j)_T & (\varphi_j, \varphi_k)_T \\ & & (\varphi_k, \varphi_k)_T \end{bmatrix}. \tag{3.44}$$

The global system matrix A may thus be computed by first computing the mass and stiffness matrices for each $T \in T_h$ and then summing the contributions from each triangle according to (3.38)–(3.40). In a corresponding way we may compute the right-hand side $\mathbf{b}^k$ (although in Chapter 4 we will show that this operation admits more efficient implementations, specially in parallel hardware). This process of computing A and $\mathbf{b}^k$ by summation is called the assembly of A and $\mathbf{b}^k$.

With notation similar to [Wathen, 1989], let n_{T_i} be the number of vertices in element $T_i \in T_h$ (i.e. local number of degrees of freedom) and n be total number of vertices in the mesh (i.e. global number of degrees of freedom). Further, let $E_{T_i} = M_{T_i} + \Delta t K_{T_i} \in \mathbb{R}^{n_{T_i} \times n_{T_i}}$ be the element coefficient matrix for each $T_i \in T_h$ and let $L_{T_i} = [l_{jk}] \in \mathbb{R}^{n_{T_i} \times n}$

$$l_{jk} := \begin{cases} 1, & \text{if } j\text{-th local node has global index } k \\ 0, & \text{otherwise} \end{cases} \tag{3.45}$$

be the boolean matrix that maps E_{T_i} into A. Then, the global assembly process can

be formulated as:

$$A = \sum_{T_i \in T_h} L_{T_i}^T E_{T_i} L_{T_i} \tag{3.46}$$

And similarly with $\mathbf{b}^k$

$$\mathbf{b}^k = \sum_{T_i \in T_h} L_{T_i}^T \mathbf{b}_{T_i}^k \tag{3.47}$$

where

$$\mathbf{b}_{T_i}^k = M_{T_i} \xi_{T_i}^k + \Delta t \mathbf{c}_{T_i} + \Delta t \mathbf{d}_{T_i} \tag{3.48}$$

is the contribution of element T_i to the RHS and ξ_{T_i}, $\mathbf{c}_{T_i}$, and $\mathbf{d}_{T_i}$ are subvectors of ξ, $\mathbf{c}$, and $\mathbf{d}$ containing only the values corresponding to the vertices of T_i.

To compute the entries in the system matrix A, one has to work with the restrictions of the basis functions φ_i, φ_j, and φ_k to the triangle $T \in T_h$. Denoting these restrictions by ψ_i^T, ψ_j^T, and ψ_k^T, we have that each ψ^T is a linear function on T that takes the value one at one vertex and vanishes at the other two vertices of T. We will refer to ψ_i^T, ψ_j^T, and ψ_k^T as the basis functions on the triangle T. If w is a linear function on T, then w has the representation

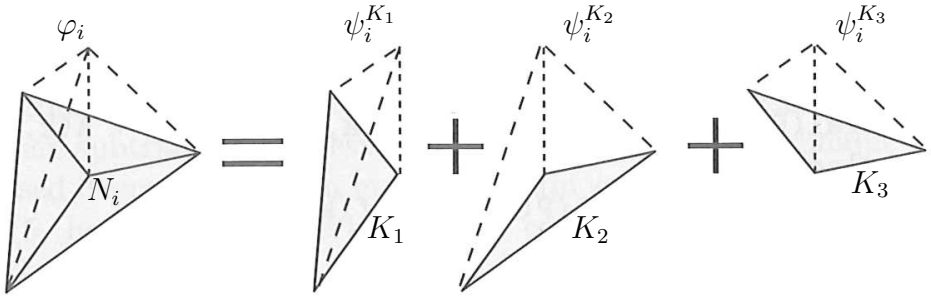
$$w(\mathbf{x}) = w(N_i)\psi_i(\mathbf{x}) + w(N_j)\psi_j(\mathbf{x}) + w(N_j)\psi_j(\mathbf{x}) \qquad \mathbf{x} \in T. \tag{3.49}$$

Figure 3.3 shows how a basis function φ_i is the summation of ψ_i^T for all the elements $T \in T_h$ that node N_i is contained in.

3.2 Solvers and preconditioners for FEM linear systems

In the previous section we introduced a model problem of a parabolic PDE over a 2D domain (P1) and derived a FEM backward Euler solution of it (P4). In this section we discuss how to efficiently solve the system of linear equations (SLE), $A\mathbf{x}^{k+1} = \mathbf{b}^k$,

Figure 3.3 Relationship between φ_i and $\psi_i^K, K \in T_h$. Adapted from [Elman et al., 2005]



arising from (P4). This SLE is characterised by a symmetric positive-definite and sparse coefficient matrix, i.e. only a very small proportion of its entries are nonzero. For the type of computational domains, Ω, that we will be using in later chapters A is also very large: N_T (also known as the number of degrees of freedom) is of order $O(10^6)$. Two alternatives will be discussed: direct and iterative methods.

Direct methods can be applied for computing the solution of $A\mathbf{x}^{k+1} = \mathbf{b}^k$ in a fixed number of steps. They are usually designed as a two-step process where, in its most basic form, the system matrix A is first factorised via Gaussian elimination (see [Golub and Van Loan, 1996])

$$LL^T\mathbf{x}^{k+1} = \mathbf{b}^k \tag{3.50}$$

where L is a $n \times n$ lower triangular matrix (for A symmetric positive definite this is known as Cholesky factorisation [Golub and Van Loan, 1996]) and later (3.50) is solved by consecutive forward and backward substitution

$$L\mathbf{y} = \mathbf{b}^k\,, \tag{3.51}$$

$$L^T\mathbf{x}^{k+1} = \mathbf{y}\,. \tag{3.52}$$

Direct methods are appealing in the context of FEM solution of PDEs involving

time derivatives where the system matrix is constant in time. In that scenario, the factorisation step is performed just once (i.e. at the beginning of the simulation) and the factors are reused to solve against multiple right-hand sides. Unfortunately, the sparsity characteristics of A do not necessarily hold for L due to a by-product of the factorisation know as matrix fill-in.

There are very effective versions of this strategy, so-called sparse elimination methods, which use sophisticated techniques to exploit sparsity in the coefficient matrix to reduce computational requirements. These include frontal methods that interleave computation of the system matrix and its factorisation (see [Irons, 1970]), and reordering strategies, which reorder the rows and columns of the coefficient matrix so that the matrix factors arising from the elimination process are made sparse to the maximum extent possible (i.e. fill-in reduction). A precise statement of the applicability and effectiveness of such methods is highly dependent on the matrix structure. Generally, direct methods work well for SLE with thousands of degrees of freedom, but they require infeasible computational resources for SLE of much larger dimension. That is, the degree of fill-in grows very quickly with n, increasing storage requirements and computational time required to perform the substitutions beyond feasibility for the values of n that we will be considering in this Thesis. In [Elman et al., 2005], direct sparse methods are considered competitive for two-dimensional partial differential equations problems but iterative methods are recommended in three dimensions.

An alternative to direct methods is the use of iterative algorithms for the solution of SLE. In this family of methods, one starts with an initial guess for $\mathbf{x}^{k+1}$, namely $\mathbf{x}^{(0)}$, and iteratively generates a sequence of approximations $\mathbf{x}^{(i)}, i = 1, 2, \ldots$, that under certain circumstances converges to the solution $\mathbf{x}^{k+1}$. One of the advantages of this approach is that one can stop the iteration as soon as $\mathbf{x}^{(i)}$ is close enough to $\mathbf{x}^{k+1}$, which in its most basic form is done by monitoring the evolution of the

residual

$$\mathbf{r}^{(i)} = A\mathbf{b}^k - \mathbf{x}^{(i)}\,, \tag{3.53}$$

under a certain norm. There exist a plethora of these methods and a complete survey is outside the scope of this work (the interested reader can refer to [Barrett et al., 1994; Elman et al., 2005; Golub and Van Loan, 1996]). In this section we will present two of such algorithms: Conjugate Gradient (Section 3.2.1) and Chebyshev Iteration (Section 3.2.2). Both belong to the family of Krylov subspace methods.

A feature of iterative methods is that they can take full advantage of the sparsity of the coefficient matrix. In particular, their storage requirements typically depend only on the number of nonzeros in the matrix and do not suffer from fill-in (since there are no matrix factorisations involved). The aim then becomes to make convergence as fast as possible. We will show later in Section 3.2.3 how convergence rate is directly related to the system matrix spectral properties and how the technique known as preconditioning can be applied to speed up convergence.

3.2.1 Conjugate gradient

The Conjugate Gradient (CG) method is an effective method for symmetric positive definite systems $A\mathbf{x} = \mathbf{b}$. Given an initial guess $\mathbf{x}^{(0)}$ (which can be a zero vector), the method proceeds by generating a sequence of iterates $\mathbf{x}^{(i)}$ in $\mathbf{x}^{(0)} + \mathcal{K}_i(A, \mathbf{r}^{(0)})$, where the Krylov subspace $\mathcal{K}_i(A, \mathbf{v})$ is defined as

$$\mathcal{K}_i(A, \mathbf{v}) := \text{span}(\mathbf{v}, A\mathbf{v}, A^2\mathbf{v}, \ldots, A^{(i-1)}\mathbf{v})\,. \tag{3.54}$$

Note that $\mathcal{K}_n = V_h$. Although the length of this sequence can become large, only a small number of vectors needs to be kept in memory.

The basic idea behind CG can be better understood by recasting the original

problem as the problem of minimising the so-called quadratic form of the SLE

$$f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T A\mathbf{x} - \mathbf{b}^T\mathbf{x}\,, \tag{3.55}$$

which is to find the solution of:

$$f'(\mathbf{x}) = A\mathbf{x} - \mathbf{b} = 0\,. \tag{3.56}$$

Note that (3.56) has a unique solution for symmetric positive-definite systems [Shewchuk, 1994]. In order to find it the algorithm takes $\mathbf{x}^{(0)}$ as an initial search point and proceeds by generating a sequence of search directions and magnitudes satisfying certain orthogonality conditions.

The iterates $\mathbf{x}^{(i)}$ are updated in each iteration by a multiple α_i of the search direction vector $\mathbf{p}^{(i)}$:

$$\mathbf{x}^{(i)} = \mathbf{x}^{(i-1)} + \alpha_i\mathbf{p}^{(i)}\,. \tag{3.57}$$

Correspondingly the residuals are updated as

$$\mathbf{r}^{(i)} = \mathbf{r}^{(i-1)} - \alpha_i A\mathbf{p}^{(i)}\,. \tag{3.58}$$

The magnitude of the update in a given direction

$$\alpha_i = \frac{\mathbf{r}^{(i-1)^T}\mathbf{r}^{(i-1)}}{\mathbf{p}^{(i)^T}A\mathbf{p}^{(i)}} \tag{3.59}$$

is chosen such that the new iterate minimises the A-norm[2] of the error among all the possible choices along that direction search

$$||\mathbf{x} - \mathbf{x}^{(i)}||_A = \min_{\alpha_i} ||\mathbf{x} - (\mathbf{x}^{(i-1)} + \alpha_i\mathbf{p}^{(i)})||_A = \min_{\mathbf{v}=\mathbf{x}^{(i-1)}+\alpha_i\mathbf{p}^{(i)}} ||\mathbf{x} - \mathbf{v}||_A\,. \tag{3.60}$$

[2]The A-norm of a vector $\mathbf{v}$ is defined as $||\mathbf{v}||_A = \mathbf{v}^T A\mathbf{v}$.

More importantly, the one-dimensional minimisation is also an i-dimensional one: $\mathbf{x}^{(i)}$ is the unique vector in the translated Krylov space $\mathbf{x}^{(0)} + \mathcal{K}_i(A, \mathbf{r}^{(0)})$ for which the A-norm of the error is minimised. See [Elman et al., 2005, Theorem 2.2] for a proof.

The first search is done in the direction of $\mathbf{r}^{(0)}$, i.e. $\mathbf{p}^{(1)} = \mathbf{r}^{(0)}$, and following search directions are updated based on the residuals

$$\mathbf{p}^{(i)} = \mathbf{r}^{(i-1)} + \beta_{i-1}\mathbf{p}^{(i-1)}\,, \tag{3.61}$$

where the choice $\beta_i = \mathbf{r}^{(i)^T}\mathbf{r}^{(i)}/\mathbf{r}^{(i-1)^T}\mathbf{r}^{(i-1)}$ ensures that $\mathbf{p}^{(i)}$ is A-orthogonal to all the previous search directions $\mathbf{p}^{(j)}$ (i.e. $\mathbf{p}^{(i)^T}A\mathbf{p}^{(j)} = \delta_{ij}$) and $\mathbf{r}^{(i)}$ is orthogonal to all the previous residuals $\mathbf{r}^{(j)}$.

The pseudocode for the preconditioned Conjugate Gradient method is given in Algorithm 1. It uses a preconditoner P which we will introduce in Section 3.2.3; for $P = I$ one obtains the unpreconditioned version of the Conjugate Gradient algorithm.

Theory

The unpreconditioned CG method constructs the i-th iterate $\mathbf{x}^{(i)}$ as an element of $\mathbf{x}^{(0)} + \text{span}\{\mathbf{r}^{(0)}, \ldots, A^{(i-1)}\mathbf{r}^{(0)}\}$ so that $||\mathbf{x}^{(i)} - \mathbf{x}||_A$ is minimised, where $\mathbf{x}$ is the exact solution of the system. This minimum is guaranteed to exist in general only if A is symmetric positive definite.

Since for a symmetric matrix, A, an orthogonal basis for the Krylov subspace $\mathcal{K}_i(A, \mathbf{r}^{(0)})$ can be constructed with only three-term recurrences, such a recurrence also suffices for generating the residuals. In the Conjugate Gradient method two coupled two-term recurrences are used; one that updates residuals using a search direction vector, and one updating the search direction with a newly computed residual. This makes the Conjugate Gradient Method quite attractive computationally.

Algorithm 1 Preconditioned conjugate gradient.

INPUT: System matrix $A \in \mathbb{R}^{n \times n}$, system RHS $\mathbf{b} \in \mathbb{R}^n$, initial guess $\mathbf{x}^{(0)} \in \mathbb{R}^n$, convergence criteria.

OUTPUT: $\mathbf{x}^{(j)} \in \mathbb{R}^n$, approximation to the solution of the system $A\mathbf{x} = \mathbf{b}$ meeting the convergence criteria after j iterations.

1. **for** $i := 1, 2, \ldots$ **until** convergence **do**
2. **solve** $P\mathbf{z}^{(i-1)} = \mathbf{r}^{(i-1)}$
3. $\rho_{i-1} := \mathbf{r}^{(i-1)^T}\mathbf{z}^{(i-1)}$
4. **if** $i = 1$ **then**
5. $\mathbf{p}^{(1)} := \mathbf{z}^{(0)}$
6. **else**
7. $\beta_{i-1} := \rho_{i-1}/\rho_{i-2}$
8. $\mathbf{p}^{(i)} := \mathbf{z}^{(i-1)} + \beta_{i-1}\mathbf{p}^{(i-1)}$
9. **endif**
10. $\mathbf{q}^{(i)} := A\mathbf{p}^{(i)}$
11. $\alpha_i := \rho_{i-1}/\mathbf{p}^{(i)^T}\mathbf{q}^{(i)}$
12. $\mathbf{x}^{(i)} := \mathbf{x}^{(i-1)} + \alpha_i\mathbf{p}^{(i)}$
13. $\mathbf{r}^{(i)} := \mathbf{r}^{(i-1)} - \alpha_i\mathbf{q}^{(i)}$
14. check convergence
15 **endfor**

There is a close relationship between the Conjugate Gradient method and the Lanczos method for determining eigensystems, since both are based on the construction of an orthogonal basis for the Krylov subspace, and a similarity transformation of the coefficient matrix to tridiagonal form. The coefficients computed during the CG iteration then arise from the LU factorization of this tridiagonal matrix. From the CG iteration one can reconstruct the Lanczos process, and vice versa; see [Golub and Van Loan, 1996]. This relationship can be exploited to obtain relevant information about the eigensystem of the (preconditioned) matrix. We will exploit this property later in Chapter 4.

Convergence

Accurate predictions of the convergence of iterative methods are difficult to make, but useful bounds can often be obtained. For the Conjugate Gradient method, the error can be bounded in terms of the spectral condition number of the preconditioned system matrix, $\kappa_2 \equiv \kappa_2(A)$, where

$$\kappa_2(A) = \frac{\lambda_{\max}(A)}{\lambda_{\min}(A)} \qquad A = A^T, \mathbf{v}^T A \mathbf{v} > 0\,. \tag{3.62}$$

and $\lambda_{\max}(A), \lambda_{\min}(A)$ are the largest and smallest eigenvalues of A, respectively.

Further, for a symmetric positive definite system $A\mathbf{x} = \mathbf{b}$ the sequence of iterates, $\mathbf{x}^{(i)}$, generated by CG can be shown to satisfy [Golub and Van Loan, 1996]

$$||\mathbf{x}^{(i)} - \mathbf{x}||_A \leq \left(\frac{\sqrt{\kappa_2} - 1}{\sqrt{\kappa_2} + 1}\right)^i ||\mathbf{x}^{(0)} - \mathbf{x}||_A\,. \tag{3.63}$$

From this relation it can be seen that the number of iterations required to reach a relative reduction of ϵ in the error is proportional to $\sqrt{\kappa_2}$.

In some cases, practical application of the above error bound is straightforward. For example, elliptic second order PDEs typically give rise to coefficient matrices A

with $\kappa_2(A) = O(h^{-2})$ [Wathen, 1989]. Hence, a number of CG iterations proportional to h^{-1} can be expected.

Other results concerning the behavior of CG have been obtained. If the extremal eigenvalues of the matrix A are well separated, then one often observes the so-called superlinear convergence [Concus et al., 1976]; that is, convergence at a rate that increases per iteration. This phenomenon is explained by the fact that CG tends to eliminate components of the error in the direction of eigenvectors associated with extremal eigenvalues first. After these have been eliminated, the method proceeds as if these eigenvalues did not exist in the given system, i.e. the convergence rate depends on a reduced system with a (much) smaller condition number.

3.2.2 Chebyshev iteration

The Chebyshev Iteration (CI) is an iterative method for the solution of nonsymmetric linear systems [Manteuffel, 1977]. CI avoids the inner products necessary for the computation of the search coefficient, α_i, in CG. This characteristic is very desirable when solving linear systems in distributed memory parallel machines, since the performance of this computational kernel tends to scale poorly with number of processors.

Like any other Krylov subspace method, CI generates a sequence of iterates $\mathbf{x}^{(i)}$ in the space

$$\mathbf{x}^{(0)} + \mathcal{K}_i(A, \mathbf{r}^{(0)}) \,, \tag{3.64}$$

that approximate the solution of the system $A\mathbf{x} = \mathbf{b}$, where $\mathbf{x}^{(0)}$ and $\mathbf{r}^{(0)}$ are the initial guess and residual respectively. Any iterate $\mathbf{x}^{(i)}$ will therefore be of the form

$$\mathbf{x}^{(i)} = \mathbf{x}^{(0)} + \sum_{j=0}^{i-1} \alpha_j A^j \mathbf{r}^{(0)} \,, \tag{3.65}$$

which can rewritten in terms of a polynomial, q_{i-1}, of degree $i-1$

$$\mathbf{x}^{(i)} = \mathbf{x}^{(0)} + q_{i-1}(A)\mathbf{r}^{(0)}\,. \tag{3.66}$$

Often this property is described in term of the residuals

$$\mathbf{r}^{(i)} = \mathbf{b} - Ax^{(i)} \tag{3.67}$$

as

$$\mathbf{r}^{(i)} = p_i(A)\mathbf{r}^{(0)}\,, \tag{3.68}$$

where p_i is a polynomial of degree i satisfying $p_i(0) = 1$. (3.68) is derived from (3.66) and (3.67):

$$\begin{aligned}
\mathbf{r}^{(i)} &= \mathbf{b} - A\mathbf{x}^{(i)} && (3.69)\\
&= \mathbf{b} - A\mathbf{x}^{(0)} - Aq_{i-1}(A)\mathbf{r}^{(0)} && (3.70)\\
&= \mathbf{r}^{(0)} - Aq_{i-1}(A)\mathbf{r}^{(0)} && (3.71)\\
&= p_i(A)\mathbf{r}^{(0)} && (3.72)
\end{aligned}$$

where $p_i(z) = 1 - zq_{i-1}(z)$ satisfies $p_i(0) = 1$.

Let $\mathcal{E}$ be an elliptic domain enveloping the matrix spectrum, $\Lambda(A)$, such that

$$\Lambda(A) \subset \mathcal{E} \wedge 0 \notin \mathcal{E}\,. \tag{3.73}$$

Denote the center of the ellipse by α, its foci by $\alpha \pm c$, and the lengths of the large and the small semi-axes by a and b. When $b = 0$, the elliptic domain turns into an interval $\mathcal{I} = [\alpha - c, \alpha + c]$.

In order to compute p_i, the Chebyshev Iteration method chooses the family of Chebyshev polynomials, T_i, shifted from $[-1, 1]$ to $\mathcal{E}$ and scaled such that $p_i(0) = 1$:

$$p_i(\zeta) = \frac{T_i((\zeta - \alpha)/c)}{T_i(-\alpha/c)}\,, \tag{3.74}$$

where

$$T_i(\zeta) := \begin{cases} \cos(i \arccos(\zeta)) & \text{if } |\zeta| \leq 1\,, \\ \cosh(i \operatorname{arccosh}(\zeta)) & \text{if } \zeta > 1\,, \\ (-1)^i T_i(-\zeta) & \text{if } \zeta < -1\,, \end{cases} \tag{3.75}$$

or equivalently

$$T_i(\zeta) = \frac{1}{2}\left[(\zeta + \sqrt{\zeta^2 - 1})^i + (\zeta - \sqrt{\zeta^2 - 1})^i\right]\,. \tag{3.76}$$

In this work, we are interested in using the method when A is symmetric positive definite and, therefore, the interval $\mathcal{I}$ lies on the positive real axis. In this case, CI is known to be optimal in the sense that it yields, for every i, the smallest i-th maximum residual if the maximum is taken over all normal matrices with spectrum on $\mathcal{I}$ (see [Gutknecht and Röllin, 2002] and references therein).

In practice, computing (3.76) on each iteration would be too costly and the classical three term recurrence for Chebyshev polynomials is used instead

$$\begin{aligned} &T_0(\zeta) = 1, \quad T_1(\zeta) = \zeta, \\ &T_{i+1}(\zeta) = 2\zeta T_i(\zeta) - T_{i-1}(\zeta), \qquad i > 1\,. \end{aligned} \tag{3.77}$$

Based on (3.77), [Gutknecht and Röllin, 2002] derive Algorithm 2 for computing the residuals $\mathbf{r}^{(i)}$ and iterates $\mathbf{x}^{(i)}$.

Convergence

In the symmetric case, for the Chebyshev Iteration we have the same error upper bound (3.63) as for the Conjugate Gradient method, provided that $\alpha \mp c$ accurately

Algorithm 2 Chebyshev iteration.

INPUT: System matrix $A \in \mathbb{R}^{n\times n}$, system RHS $\mathbf{b} \in \mathbb{R}^n$, initial guess $\mathbf{x}^{(0)} \in \mathbb{R}^n$, α and c such that $\Lambda(A) \subset [\alpha - c, \alpha + c]$, convergence criteria.

OUTPUT: $\mathbf{x}^{(j)} \in \mathbb{R}^n$, approximation to the solution of the system $A\mathbf{x} = \mathbf{b}$ meeting the convergence criteria after j iterations.

1. $\eta := -\frac{\alpha}{c}$
2. $\mathbf{r}^{(-1)} = \mathbf{x}^{(-1)} := (0, \ldots, 0)$'
3. $\mathbf{r}^{(0)} := \mathbf{b} - A\mathbf{x}^{(0)}$
4. $\beta_{-1} := 0$
5. $\beta_0 := \frac{c}{2\eta} = -\frac{c^2}{2\alpha}$
6. $\gamma_0 := -\alpha$
7. **for** $i := 0, 1, \ldots$ **until** convergence **do**
8. $\quad \beta_{i-1} := \frac{cT_{i-1}(\eta)}{2T_i(\eta)} = \left(\frac{c}{2}\right)^2 \frac{1}{\gamma_{i-1}}$, **if** $i \geq 2$
9. $\quad \gamma_i := \frac{cT_{i+1}(\eta)}{2T_i(\eta)} = -(\alpha + \beta_{i-1})$, **if** $i \geq 1$
10. $\quad \mathbf{x}^{(i+1)} := -(\mathbf{r}^{(i)} + \alpha\mathbf{x}^{(i)} + \beta_{i-1}\mathbf{x}^{(i-1)})\gamma_i^{-1}$
11. $\quad \mathbf{r}^{(i+1)} := (A\mathbf{r}^{(i)} - \alpha\mathbf{r}^{(i)} - \beta_{i-1}\mathbf{r}^{(i-1)})\gamma_i^{-1}$
12. $\quad$ check convergence
13 **endfor**

approximate $\lambda_{\min}$, $\lambda_{\max}$. There is a severe penalty for overestimating or underestimating $\Lambda(A)$. For example, if in the symmetric case $\lambda_{\max}$ is underestimated, then the method may diverge; if it is overestimated then convergence may be very slow. This implies that one needs fairly accurate bounds on the spectrum of A for the method to be effective. Computing these bounds for systems made of millions of degrees of freedom can be as challenging as the solution of the system itself. In Chapter 4 we will explore the relationship between CG and the Lanczos algorithm for solution of eigensystems in order to overcome this issue.

Unlike other Krylov subspace methods, CI does not exhibit superlinear convergence [Barrett et al., 1994]. This is an important drawback since, under certain circumstances, the larger number of iterations when compared to CG makes up for the reduction in time per iteration (coming from the absence of inner products), yielding shorter total execution time for CG. We will explore this question further in Chapter 4.

3.2.3 Preconditioning

We recall from (3.63) that the number of iterations required by a Krylov subspace method in order to find an approximation $\mathbf{x}^{(n)}$ to the solution of the linear system $A\mathbf{x} = \mathbf{b}$ such that $||\mathbf{r}^{(n)}||$ goes below a given tolerance is proportional to the spectral condition number of the system $\kappa_2(A)$. Note that most of the time this value is not easily accessible and one has to learn about the spectral properties of a system by analysing its convergence patterns.

The basic idea behind preconditioning is to construct a matrix (or a linear operator), P, that approximates the coefficient matrix A well while requiring little work to apply the action of the inverse of P, i.e. to compute $P^{-1}\mathbf{v}$ for a given $\mathbf{v}$. One may then think of solving

$$P^{-1}A\mathbf{x} = P^{-1}\mathbf{b} \tag{3.78}$$

instead of $A\mathbf{x} = \mathbf{b}$, which obviously yields the same solution. If P is a good approximation of A, then it might be expected that a given Krylov subspace method will be more rapidly convergent for the preconditioned system than for the original one (i.e. $\kappa_2(P^{-1}A) \ll \kappa_2(A)$), and the overall computational work may be significantly reduced. It is important to note the tradeoff between cost of applying P^{-1} and iteration reduction. In Chapters 5 and 6 we will identify situations where a simpler preconditioner yields a shorter total execution time since the matrix it acts upon is not too badly conditioned.

An important aspect to note is that $P^{-1}A$ is not formed explicitly (except for some trivial cases), since one does not have guarantee of the product being symmetric even when both P and A are; this would render CG unusable. Even when symmetry is known to hold or when nonsymmetric solvers, like CI, are chosen, an explicitly formed $P^{-1}A$ is likely to be dense despite A being sparse.

Another important aspect of preconditioning linear systems arising from FEM discretisation of PDEs is that the system matrix condition number is often an inverse function of the maximum edge length, h. For second order elliptic problems, it can be shown that [Wathen, 1989]

$$\kappa_2(A) = O(h^{-2}). \tag{3.79}$$

In this case, if one had to use finer domain discretisations in order to reduce the error of the FEM approximation, typically $O(h)$, one would find that the computational cost grows superlinearly. That is, for a finer discretisation there is not only more work in carrying out a single linear solve iteration (due to the increase in degrees of freedom), but also more iterations are required for convergence. It would be ideal if the number of iterations required to satisfy the stopping criterion did not grow under mesh refinement, so that the computational work would grow linearly with the dimension of the discrete system. In Chapter 6 we develop discretisation

independent preconditioners for the bidomain equations.

Although reducing the system condition number through preconditioning will certainly decrease the number of iterations required, this will also happen when the condition number stays constant but the eigenvalues of the system get clustered together by the action of the preconditioner. Let $(\lambda_j, \mathbf{z}_j)$ be the n eigenpairs of A such that $A\mathbf{z}_j = \lambda_j \mathbf{z}_j$ and express $\mathbf{r}^{(0)}$ in the basis of $\{\mathbf{z_j}\}$:

$$\mathbf{r}^{(0)} = \sum_j \gamma_j \mathbf{z}_j \,. \tag{3.80}$$

(3.68) can be then rewritten as [van der Vorst, 2003]:

$$\mathbf{r}^{(i)} = p_i(A)\mathbf{r}^{(0)} = \sum_j \gamma_j p_i(\lambda_j)\mathbf{z}_j \,. \tag{3.81}$$

A polynomial of degree i has i roots, i.e. i eigenvalues λ_j such that $p_i(\lambda_j) = 0$. Therefore if the action of the preconditioner results in eigenvalues of multiplicity greater than one (i.e. $\lambda_i = \lambda_k, i \neq k$) or very close to each other (such that $p(\lambda_i) = 0 \rightarrow p(\lambda_k) \simeq 0, i \neq k$), CG will convergence at a faster rate than for the original system [Shewchuk, 1994].

A plethora of preconditioning techniques have been developed for the acceleration of Krylov subspace methods. It is not our intention to survey them here (the interested reader can refer to [Benzi, 2002] and references therein). However we will introduce now the techniques used in later Chapters. The preconditioners presented here are purely algebraic in the sense that no knowledge of the physics behind the linear system being solved is taken into account. In Chapters 5 and 6, we will present problem specific preconditioners that take advantage of the matrix structure in order to reduce total execution time and achieve mesh-independent convergence.

Diagonal scaling

The choice of $P = \text{diag}(A)$ leads to diagonal scaling, also known as Jacobi preconditioning. Applying P^{-1} is trivial since it only requires n divisions. However, the action of P^{-1} only reduces the condition number of the system by a small constant, independent of h for FEM discretisations of PDEs [Elman et al., 2005].

Incomplete factorisation

We saw earlier in this section that the main reason why direct methods become unfeasible for large 3D FEM problems is the large amount of nonzeros appearing in the matrix factorisation (i.e. matrix fill-in). However, for any matrix available in triangular factored form, its corresponding linear system can be easily solved by forward and backward substitution. In order to take advantage of this while overcoming the fill-in issue, a generally applicable class of algebraic preconditioners has been developed based on the concept of incomplete $A = LU$ factorisation ($U = L^T$ for symmetric matrices).

The basic idea is to preselect some sparsity pattern outside which any nonzero entries arising in the factors L or U are dropped, either based on position or magnitude of the coefficient. The key issue becomes to choose such a pattern so that the preconditioner is not too dense and therefore cheap to apply, but not too sparse so its spectrum still resembles the original. A common choice is not to allow any fill-in and only consider the entries in the factors that were nonzero in A, namely incomplete LU(0) factorisation or ILU(0). For symmetric matrices this method is known as Incomplete Cholesky(0) or IC(0) and is presented in Algorithm 3. Note that this is the algorithm in its simplest form. Modern implementations available in optimised linear algebra packages use algorithmical improvements (such as loop unrolling, tiling, vectorisation) in order to improve performance.

Algorithm 3 Incomplete Cholesky factorisation with no fill-in.

INPUT: Matrix $A = [a_{ij}] \in \mathbb{R}^{n \times n}$.

OUTPUT: Triangular matrix $L = [l_{ij}] \in \mathbb{R}^{n \times n}$ such that $A = LL^T$.

1. **for** $i := 1$ **until** n **do**
2. $\quad m := min\{k : a_{ik} \neq 0\}$
3. $\quad$ **for** $j := m$ **until** $i - 1$ **do**
4. $\quad\quad$ **if** $a_{ij} \neq 0$ **then**
5. $\quad\quad\quad l_{ij} := (a_{ij} - \sum_{k=m}^{j-1} l_{ik} l_{jk})/l_{jj}$
6. $\quad\quad$ **endif**
7. $\quad$ **endfor**
8. $\quad l_{ii} := \left(a_{ii} - \sum_{k=m}^{i-1} l_{ik} l_{ik}\right)^{1/2}$
9. **endfor**

Domain decomposition

Another way of relaxing factorisation-based preconditioners is to consider a partition of the original problem, Ω, into a set of subproblems $\Omega_k, k = 1, \ldots, n_d$, and then restrict the factorisation to each of the subproblems or subdomains (i.e. domain decomposition). This can be better understood in terms of FEM solution of PDEs: consider a computational domain made of n nodes and partition it into n_d subdomains made of, approximately, $\frac{n}{n_d}$ nodes. One could now consider the FEM solution of each of those subdomains without taking into account the dependencies among them. Such a FEM linear system (assuming the appropriate numbering of the variables) would look like

$$A_\Omega = \begin{bmatrix} A_{\Omega_1} & & & 0 \\ & A_{\Omega_2} & & \\ & & \ddots & \\ 0 & & & A_{\Omega_{n_d}} \end{bmatrix}, \qquad A_{\Omega_k} \in \mathbb{R}^{\frac{n}{n_d} \times \frac{n}{n_d}}, \tag{3.82}$$

for a partition of Ω made of non-overlapping subdomains Ω_k. One could construct a preconditioner P based on A_Ω instead of A. For instance, one option is to compute the action of $P^{-1} = A_\Omega^{-1}$ by means of $IC(0)$ factorisations of each subblock. This method is known in the literature as the block Jacobi method, with $IC(0)$ in each block, i.e. BJACOBI($IC(0)$) [van der Vorst, 2003]. This method is very attractive for its good parallelisation properties: by taking n_d equal to the number of processors involved in a simulation, p, a communication free preconditioner is obtained. The main drawback of this technique is that as p increases the quality of the preconditioner decreases due to more and more interdomain dependencies being ignored. In some cases, this leads to an increase in the number of iterations with p.

Another option is to consider overlapping domains Ω_k, this is known in the literature as the additive Schwarz method (ASM), which combined with $IC(0)$ gives the ASM($IC(0)$) that will be used in later chapters. This approach leads to overlapping blocks A_{Ω_k} and therefore communication among domains is necessary, although to a lesser extend that in full matrix $IC(0)$.

Multigrid methods

Multigrid methods can be very effective linear system solvers for elliptic PDEs. Correspondingly, a single multigrid cycle can be an extremely effective preconditioner even for problems where it is not clear how to construct convergent multigrid iterations [Elman et al., 2005].

A multigrid method works as follows. At each grid level, a smoother is applied to the system, which serves to resolve the high-frequency error on that level. The improved iterate is then projected onto a coarser grid, the smoother is applied again, and the process continues until the coarsest level is reached. The coarsest level is generally chosen to be a size that is reasonable to solve directly, and the goal is to eliminate a significant part of the error by the time this coarsest level is reached.

The solution to the coarse grid solve is then interpolated, level by level, back up to the finest grid level, applying the smoother again at each level. A simple cycle down and up the grid is referred to as a V-cycle. To obtain good convergence, the smoother and the coarse-grid correction process must complement each other to remove all components of the error. If the components of the V-cycle are defined properly, the result is a method that uniformly damps all error frequencies with a computational cost that depends only linearly on the problem size.

A multigrid method has two phases: the setup phase and the solve phase. The setup phase consists of defining the coarse grids, interpolation operators, and coarse-grid operators for each of the coarse-grid levels. The solve phase consists of performing the multilevel cycles (i.e., iterations) until the desired convergence is obtained.

There are two basic multigrid approaches: geometric and algebraic. In geometric multigrid, the geometry of the problem is used to define the various multigrid components. Algebraic multigrid (AMG) [Falgout, 2006] was introduced as a method for solving linear systems based on multigrid principles, but in a way that requires no explicit knowledge of the problem geometry. The AMG method determines smoothers, projection operators, and interpolation operators based solely on the matrix coefficients.

3.2.4 Software for the solution of FEM linear systems

The solution of large and sparse systems of linear equations is ubiquitous in science and engineering. For that reason, the development of robust and efficient implementations of the algorithms described in this section is an active field of research. There exist a large number of software packages implementing the most commonly used Krylov subspace methods and preconditioning techniques. We will now briefly introduce two that are widely used by the FEM community.

PETSc

The Portable, Extensible Toolkit for Scientific Computation (PETSc) is a suite of data structures and routines for the efficient parallel solution of scientific applications modelled by partial differential equations. It employs the Message Passing Interface (MPI) standard for all message-passing communication. PETSc is designed to link against optimised versions of BLAS and LAPACK for efficient sequential dense algebra operations. In machines where they are not available a reference implementation is provided. The current version of PETSc is 3.1; released March 25, 2010.

PETSc is intended for use in large-scale application projects, many ongoing computational science projects are built around the PETSc libraries. In our case, PETSc was adopted during early Chaste development stages because of its parallel sparse linear algebra capabilities. PETSc includes a large suite of parallel linear and nonlinear system solvers that are easily used in application codes written in C, C++, Fortran and Python. Its design allows users to interact with the library at different levels of detail. In this most basic form, applications can use PETSc's components such as linear solvers or preconditioners as black boxes. At an intermediate level, users can optimise certain algorithm configurations. Advanced users can integrate their own numerical methods following a well-defined API. This approach proved successful in Chapter 5 and 6, where we were able to implement the novel preconditioning techniques presented and scale them to thousands of processors. PETSc provides several other features needed within parallel application code, such as simple parallel matrix and vector assembly routines that allow the overlap of communication and computation.

In PETSc, matrices and vectors (and therefore linear systems) are partitioned and distributed row-wise. In a parallel environment consisting of p processors, a

linear system $A\mathbf{x} = \mathbf{b}$ with $A \in \mathbb{R}^{n\times n}$ and $\mathbf{x}, \mathbf{b} \in \mathbb{R}^n$ will be distributed as

$$\begin{pmatrix} A_0 \\ A_1 \\ \vdots \\ A_{p-1} \end{pmatrix} \begin{pmatrix} \mathbf{x}_0 \\ \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_{p-1} \end{pmatrix} = \begin{pmatrix} \mathbf{b}_0 \\ \mathbf{b}_1 \\ \vdots \\ \mathbf{b}_{p-1} \end{pmatrix} \tag{3.83}$$

with $A_i \in \mathbb{R}^{\frac{n}{p}\times n}$ and $\mathbf{x}_i, \mathbf{b}_i \in \mathbb{R}^{\frac{n}{p}}$ (assuming $n = kp$, $k \in \mathbb{N}$ for simplicity). A_i, $\mathbf{x}_i$, and $\mathbf{b}_i$ are the sub-blocks assigned to processor i.

Finally, PETSc provides interfaces for several other parallel sparse linear algebra packages: HYPRE, Trilinos, SuperLU, etc. This allows access to a wide range of algorithms from a unified set of data structures. In our case, we take advantage of PETSc's interface to the package HYPRE in order to use some of its preconditioners. In this work, we will use PETSc's implementation of CG, CI, sequential IC(0), parallel BJACOBI(IC(0)), and parallel ASM(IC(0)) without changes to the standard configuration.

hypre

The Parallel High Performance Preconditioners (hypre) is a library of high performance preconditioners and solvers for the solution of large, sparse linear systems of equations on massively parallel computers. The hypre library was created with the primary goal of providing users with advanced parallel preconditioners. The library features parallel multigrid solvers for both structured and unstructured grid problems. Hypre uses the MPI standard for all message-passing communication and, similarly to PETSc, relies on a system install of BLAS for dense linear algebra operations. PETSc has an interface to call hypre preconditioners.

In this work, parallel AMG preconditioning and parallel $ILU(0)$ factorisation is performed with the BoomerAMG and EUCLID algorithms found in version 2.4.0b

of the HYPRE library (through its PETSc interface). BoomerAMG is configured with the following options:

```
-pc_hypre_type boomeramg
-pc_hypre_boomeramg_max_iter 1
-pc_hypre_boomeramg_strong_threshold 0.0
```

We choose this AMG implementation for its maturity in terms of parallel performance [Baker et al., 2010] and for having being successfully applied to the solution of the bidomain equations before [Plank et al., 2007].

3.3 Finite element solution of the bidomain equations

In Chapter 2, we presented a derivation of the bidomain equations of cardiac electrophysiology. In the current section we will apply the FEM framework described in Section 3.1 for the discretisation and solution of the bidomain equations with and without surrounding medium.

3.3.1 The bidomain equations on cardiac tissue

Formulation

Let Ω denote the region occupied by the cardiac tissue. Let the unknown electrical potential fields in the intracellular and extracellular spaces, defined throughout Ω, be ϕ_i and ϕ_e respectively, and let V denote the transmembrane voltage, $V = \phi_i - \phi_e$.

The *parabolic-parabolic* (PP) form of the bidomain equations [Keener and Sneyd,

1998] is

$$\begin{aligned} \nabla \cdot (\sigma_i \boldsymbol{\nabla} \phi_i) &= \chi \left(\mathcal{C} \frac{\partial V}{\partial t} + I_{\text{ion}}(\mathbf{u}, V) \right) + I_i^{(\text{vol})} && (3.84) \\ \nabla \cdot (\sigma_e \boldsymbol{\nabla} \phi_e) &= -\chi \left(\mathcal{C} \frac{\partial V}{\partial t} + I_{\text{ion}}(\mathbf{u}, V) \right) + I_e^{(\text{vol})} && (3.85) \\ \frac{\partial \mathbf{u}}{\partial t} &= \mathbf{f}(\mathbf{u}, V) \end{aligned}$$

where σ_i is the intracellular conductivity tensor, σ_e the extracellular conductivity tensor, χ is the surface area-to-volume ratio and $\mathcal{C}$ is the membrane capacitance per unit area. The vector $\mathbf{u}$ contains cell-level variables, such as ionic concentrations and membrane gating variables, and $I_{\text{ion}} \equiv I_{\text{ion}}(\mathbf{u}, V)$ is the ionic current per unit surface area. Functional forms for I_{ion} and $\mathbf{f}$ are determined by an electrophysiological cell model (see Section 2.2.1 for more details). The source terms $I_i^{(\text{vol})}$ and $I_e^{(\text{vol})}$ are the intracellular and extracellular stimuli per unit volume.

The *parabolic-elliptic* (PE) form of the bidomain equations is obtained by replacing (3.85) with the sum of (3.84) and (3.85):

$$\begin{aligned} \chi \left(\mathcal{C} \frac{\partial V}{\partial t} + I_{\text{ion}}(\mathbf{u}, V) \right) - \nabla \cdot (\sigma_i \boldsymbol{\nabla} (V + \phi_e)) &= -I_i^{(\text{vol})} && (3.86) \\ \nabla \cdot (\sigma_i \boldsymbol{\nabla} V + (\sigma_i + \sigma_e) \boldsymbol{\nabla} \phi_e) &= I_{\text{total}}^{(\text{vol})} && (3.87) \\ \frac{\partial \mathbf{u}}{\partial t} &= \mathbf{f}(\mathbf{u}, V) && (3.88) \end{aligned}$$

where $I_{\text{total}}^{(\text{vol})} = I_i^{(\text{vol})} + I_e^{(\text{vol})}$.

Appropriate boundary conditions for both formulations are zero flux of ϕ_i and ϕ_e over the boundary of the domain:

$$\begin{aligned} \mathbf{n} \cdot (\sigma_i \boldsymbol{\nabla} (V + \phi_e)) &= 0, \qquad \text{on } \partial\Omega && (3.89) \\ \mathbf{n} \cdot (\sigma_e \boldsymbol{\nabla} \phi_e) &= 0, \qquad \text{on } \partial\Omega && (3.90) \end{aligned}$$

where $\mathbf{n}$ is the outward-facing unit normal. Note that these equations only define ϕ_e up to a constant and therefore form a singular system. As such, there is a compatibility condition, $\int_\Omega I_{\text{total}}^{(\text{vol})} \, \mathrm{d}^3\mathbf{x} = 0$, which must be satisfied for solutions to exist [Pathmanathan et al., 2010]. We take $I_e^{(\text{vol})} = -I_i^{(\text{vol})}$, which ensures this condition is met.

Finite element solution

The finite element method can be used for the spatial discretisation of the bidomain equations. In FEM, the original domain is discretised into a finite set of nodes and elements and the original system of PDEs is approximated with a system of linear equations defining the solution of the problem at those nodes. We will now describe the FEM linear system that must be solved at each timestep. We use a coupled approach in which the two unknowns (i.e. (ϕ_i, ϕ_e) or (V, ϕ_e), depending on the formulation) are computed together [Southern et al., 2009; Sundnes et al., 2006; Whiteley, 2006] and choose a semi-implicit time-discretisation where the conduction term is treated implicitly but the reaction term explicitly. This is a compromise between the numerical instability of a fully explicit method and computational burden of a fully implicit approach. See [Whiteley, 2006] for a comprehensive discussion on the three approaches cited.

The decoupling of the reactive and diffusive parts of the bidomain equations achieved by means of the semi-implicit time discretisation used in this Thesis is equivalent to the application of operator splitting to equations (3.88) and (3.86)–(3.87). This treatment of PDE systems often eases their numerical solution but introduces a splitting error that is larger the more tightly coupled the systems are and can also change over time depending on the dynamics of the system. Due to the high level of complexity of the systems studied, it is difficult to perform a detailed mathematical analysis of the accuracy and stability of operator splitting

techniques [Linge et al., 2009]. [Schroll et al., 2007] analysed the splitting error in monodomain systems and concluded that a) the order in which the reaction and diffusion steps are taken is irrelevant, and b) the convergence rate of the first-order numerical method analysed is not affected by operator splitting. [Sundnes et al., 2005] presented first-order and second-order accurate operator splitting methods for the bidomain equations (i.e. Godunov and Strang splitting) and analysed them numerically. They concluded that in the case of realistic cell models (e.g. [Winslow et al., 1999]) Strang splitting fails to achieve second-order convergence. The operator splitting method used in this Thesis is equivalent to the Godunov splitting presented in [Sundnes et al., 2005], the reader can refer to this publication for detailed analysis and references.

Let V^m denote the voltage at time t_m, and similarly with other variables. Using a semi-implicit time-discretisation, the weak statement corresponding to the parabolic-elliptic formulation (3.86)–(3.90), found by integrating over the domain, and using the divergence theorem and the boundary conditions, is: given V^n, find $V^{n+1} \in H^1(\Omega)$ and $\phi_e^{n+1} \in H^1(\Omega)$ such that

$$\begin{aligned} & \frac{\chi\mathcal{C}}{\Delta t}\int_\Omega V^{n+1} v \,\mathrm{d}^3\mathbf{x} + \int_\Omega \left(\sigma_i \boldsymbol{\nabla}(V^{n+1} + \phi_e^{n+1}\right) \cdot \boldsymbol{\nabla} v \,\mathrm{d}^3\mathbf{x} \\ = \; & \frac{\chi\mathcal{C}}{\Delta t}\int_\Omega V^n v \,\mathrm{d}^3\mathbf{x} - \int_\Omega \left(I_i^{(\mathrm{vol})} + \chi I_{\mathrm{ion}}(\mathbf{u}, V^n)\right) v \,\mathrm{d}^3\mathbf{x} \qquad \forall v \in H^1(\Omega), \end{aligned} \tag{3.91}$$

$$\int_\Omega \left(\sigma_i \boldsymbol{\nabla} V^{n+1} + (\sigma_i + \sigma_e)\,\boldsymbol{\nabla}\phi_e^{n+1}\right) \cdot \boldsymbol{\nabla} v \,\mathrm{d}^3\mathbf{x} = 0 \qquad \forall v \in H^1(\Omega). \tag{3.92}$$

For the finite element discretisation of the *parabolic-elliptic* formulation (3.86)–(3.90), we choose a set of basis functions $\psi_1, \psi_2, \ldots, \psi_N$, where N is the number of nodes in the finite element mesh, and write $V^n(x) = \sum V_k^n \psi_k(x)$, $V_k^n \in \mathbb{R}$, and similarly $\phi_e^n(x) = \sum \Phi_k^n \psi_k(x)$. We define $\mathbf{V}^n = (V_1^n, V_2^n, \ldots, V_N^n)$ and $\boldsymbol{\Phi}_e^n =$

$(\Phi_1^n, \Phi_2^n, \ldots, \Phi_N^n)$. The finite element problem is to find $\mathbf{V^{n+1}}$ and $\mathbf{\Phi_e^{n+1}}$ such that

$$\frac{\chi\mathcal{C}}{\Delta t}M\mathbf{V}^{n+1} + K[\sigma_i]\mathbf{V}^{n+1} + K[\sigma_i]\mathbf{\Phi}_e^{n+1} = \frac{\chi\mathcal{C}}{\Delta t}M\mathbf{V}^n + \mathbf{c}^n$$

$$K[\sigma_i]\mathbf{V}^{n+1} + K[\sigma_i + \sigma_e]\mathbf{\Phi}_e^{n+1} = 0$$

where

$$\begin{aligned} M_{jk} &= \int_\Omega \psi_j\psi_k \,\mathrm{d}^3\mathbf{x} \\ \left(K[\sigma]\right)_{jk} &= \int_\Omega \boldsymbol{\nabla}\psi_j \cdot (\sigma\boldsymbol{\nabla}\psi_k) \,\mathrm{d}^3\mathbf{x} \\ c_j^n &= -\int_\Omega \left(I_i^{(\mathrm{vol})}(t_{n+1}) + \chi I_{\mathrm{ion}}(\mathbf{u}^n, V^n)\right)\psi_j \,\mathrm{d}^3\mathbf{x}. \end{aligned}$$

Overall, we have the $2N$ equations

$$\begin{pmatrix} \frac{\chi\mathcal{C}}{\Delta t}M + K[\sigma_i] & K[\sigma_i] \\ K[\sigma_i] & K[\sigma_i + \sigma_e] \end{pmatrix} \begin{pmatrix} \mathbf{V}^{n+1} \\ \mathbf{\Phi}_e^{n+1} \end{pmatrix} = \begin{pmatrix} \frac{\chi\mathcal{C}}{\Delta t}M\mathbf{V}^n + \mathbf{c} \\ \mathbf{0} \end{pmatrix}. \tag{3.93}$$

If instead we choose to discretise the parabolic-parabolic formulation—(3.84) and (3.85) with $I_e^{(\mathrm{vol})} = -I_i^{(\mathrm{vol})}$—the corresponding linear system to be solved is

$$\begin{pmatrix} \frac{\chi\mathcal{C}}{\Delta t}M + K[\sigma_i] & -\frac{\chi\mathcal{C}}{\Delta t}M \\ -\frac{\chi\mathcal{C}}{\Delta t}M & \frac{\chi\mathcal{C}}{\Delta t}M + K[\sigma_e] \end{pmatrix} \begin{pmatrix} \mathbf{\Phi}_i^{n+1} \\ \mathbf{\Phi}_e^{n+1} \end{pmatrix} = \begin{pmatrix} \frac{\chi\mathcal{C}}{\Delta t}M\mathbf{V}^n + \mathbf{c} \\ -\frac{\chi\mathcal{C}}{\Delta t}M\mathbf{V}^n - \mathbf{c} \end{pmatrix}. \tag{3.94}$$

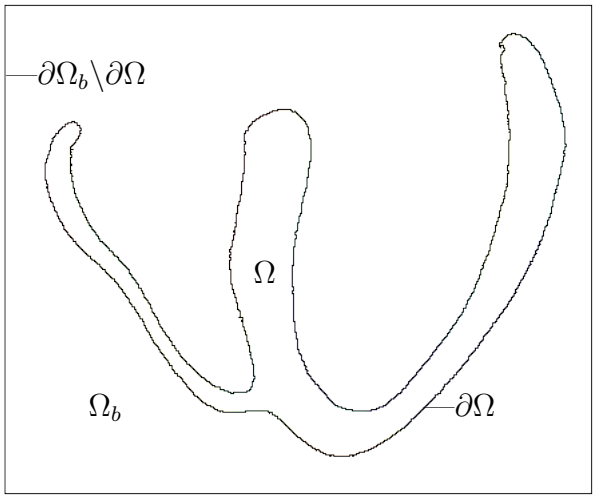
3.3.2 The bidomain equations including representation of the medium surrounding cardiac tissue

Formulation

Next, we consider tissue contained in a surrounding medium where only extracellular potential is defined. We suppose there are two disjoint domains: the tissue Ω with

boundary $\partial\Omega$ and the bath Ω_b with boundary $\partial\Omega_b$: $\partial\Omega \cap \partial\Omega_b \neq \emptyset$ (i.e. $\partial\Omega$ is an interface between both domains — see Figure 3.4 for an example). In this problem ϕ_i, and therefore V, is only defined in Ω, but ϕ_e is defined throughout $\Omega \cup \Omega_b$.

Figure 3.4 Different domains and boundaries in a model of cardiac tissue contained in a conductive bath.



The problem to be solved is: find $V \in H^1(\Omega)$ and $\phi_e \in H^1(\Omega \cup \Omega_b)$ satisfying

$$\chi\left(\mathcal{C}\frac{\partial V}{\partial t} + I_{\text{ion}}\right) - \nabla \cdot (\sigma_i \boldsymbol{\nabla}\phi_i) = -I_i^{(\text{vol})}, \text{ in } \Omega \tag{3.95}$$

$$\nabla \cdot (\sigma_i \boldsymbol{\nabla}\phi_i + \sigma_e \boldsymbol{\nabla}\phi_e) = 0, \quad \text{in } \Omega \tag{3.96}$$

$$\nabla \cdot (\sigma_b \boldsymbol{\nabla}\phi_e) = 0, \quad \text{in } \Omega_b \tag{3.97}$$

$$\frac{\partial \mathbf{u}}{\partial t} = \mathbf{f}(\mathbf{u}, V), \text{in } \Omega$$

(note: we have used ϕ_i here instead of $V + \phi_e$ in order to simplify the equations), where σ_b is the bath conductivity, with boundary conditions

$$\mathbf{n} \cdot (\sigma_i \boldsymbol{\nabla}\phi_i) = 0, \quad \text{on } \partial\Omega \tag{3.98}$$

$$\mathbf{n} \cdot (\sigma_b \boldsymbol{\nabla}\phi_e) = I^{(\text{E})}, \quad \text{on } \partial\Omega_b \backslash \partial\Omega \tag{3.99}$$

(where $\partial\Omega_b \backslash \partial\Omega$ represents the external bath boundary). Here $I^{(\mathrm{E})}$ is a stimulus current per unit area representing an external current flowing into the domain and causing an extracellular shock on the tissue surface. Note that, since the problem is stated without any Dirichlet boundary conditions on ϕ_e, it is only defined up to a constant. Compatibility conditions again ensue and $\int_{\partial\Omega_b \backslash \partial\Omega} I^{(\mathrm{E})} \, \mathrm{d}S$ must be equal to zero for solutions to exist.

An interface boundary condition is also required: it is

$$\left((\sigma_e \boldsymbol{\nabla} \phi_e) \cdot \mathbf{n}\right) \big|_{\Omega \to \partial\Omega} + \left((\sigma_b \boldsymbol{\nabla} \phi_e) \cdot \mathbf{n}\right) \big|_{\Omega_b \to \partial\Omega} = 0, \text{ on } \partial\Omega \tag{3.100}$$

(where $\big|_{\Omega \to \partial\Omega}$ denotes the limit as $\mathbf{x} \in \Omega$ tends to $\partial\Omega$), and $\big|_{\Omega_b \to \partial\Omega}$ denotes the limit as $\mathbf{x} \in \Omega_b$ tends to $\partial\Omega$). This boundary condition does not need to be explicitly enforced since it naturally arises in the weak form.

Finite element solution

Here we write the weak form before time-discretisation, for clarity. The first equation of the weak form is found by multiplying (3.95) by $v \in H^1(\Omega)$ and integrating using the divergence theorem and the boundary conditions. The second equation of the weak form is found analogously after multiplying (3.96) and also (3.97) (essentially one equation over the whole domain $\Omega \cup \Omega_b$) by $w \in H^1(\Omega \cup \Omega_b)$. The weak problem is: find $V \in H^1(\Omega)$ and $\phi_e \in H^1(\Omega \cup \Omega_b)$ satisfying:

$$\int_\Omega \chi \mathcal{C} \frac{\partial V}{\partial t} v \, \mathrm{d}^3\mathrm{x} + \int_\Omega (\sigma_i \boldsymbol{\nabla} \phi_i) \cdot \boldsymbol{\nabla} v \, \mathrm{d}^3\mathrm{x} + \int_\Omega \left(\chi I_{\mathrm{ion}} + I_i^{(\mathrm{vol})} \right) v \, \mathrm{d}^3\mathrm{x} = 0 \quad \forall v \in H^1(\Omega) \tag{3.101}$$

and

$$\int_\Omega (\sigma_i \boldsymbol{\nabla} \phi_i + \sigma_e \boldsymbol{\nabla} \phi_e) \cdot \boldsymbol{\nabla} w \, \mathrm{d}^3\mathrm{x} + \int_{\Omega_b} (\sigma_b \boldsymbol{\nabla} \phi_e) \cdot \boldsymbol{\nabla} w \, \mathrm{d}^3\mathrm{x} - \int_{\partial\Omega_b \backslash \partial\Omega} I^{(\mathrm{E})} w \, \mathrm{d}S = 0 \quad \forall w \in H^1(\Omega \cup \Omega_b) \tag{3.102}$$

Note that use of the divergence theorem after integrating (3.96) and (3.97) introduces the following boundary integrals in (3.102)

$$-\int_{\partial\Omega} w\,(\sigma_e \boldsymbol{\nabla}\phi_e)\cdot\mathbf{n}\big|_{\Omega\to\partial\Omega}\,\mathrm{d}S - \int_{\partial\Omega} w\,(\sigma_b \boldsymbol{\nabla}\phi_e)\cdot\mathbf{n}\big|_{\Omega_b\to\partial\Omega}\,\mathrm{d}S,$$

but this vanishes due the interface condition (3.100) with the continuity of w across the interface.

For the finite element discretisation, let us suppose the first N nodes are contained in Ω or on the boundary $\partial\Omega$, and the next M nodes are in Ω_b or on the boundary $\partial\Omega_b\backslash\partial\Omega$. The corresponding basis functions satisfy

$$\psi_1,\ldots,\psi_N,\underbrace{\psi_{N+1},\ldots,\psi_{N+M}}_{=0 \text{ in } \Omega}.$$

We can write $V=\sum_{j=1}^{N} V_j\psi_j$, which has to be understood to apply only for $\mathbf{x}\in\overline{\Omega}$ (i.e. the closure of Ω), and $\phi_e=\sum_{j=1}^{N+M}\Phi_j\psi_j$. Let $\mathbf{V}=(V_1,\ldots,V_N)$, $\boldsymbol{\Phi}=(\Phi_1,\ldots,\Phi_{N+M})$ and define $\boldsymbol{\Phi}_{(1)}=(\Phi_1,\ldots,\Phi_N)$ and $\boldsymbol{\Phi}_{(2)}=(\Phi_{N+1},\ldots,\Phi_{N+M})$.

The final finite element linear system will be of size $2N+M$. The first N equations are obtained by setting $v=\psi_1,\ldots,\psi_N$ in (3.101). The remaining $N+M$ equations are obtained by setting $w=\psi_1,\ldots,\psi_{N+M}$ in (3.102).

Let us define $\mathcal{K}$ as the $(N+M)\times(N+M)$ stiffness matrix:

$$\mathcal{K}_{jk}=\int_{\Omega}((\sigma_i+\sigma_e)\boldsymbol{\nabla}\psi_k)\cdot\boldsymbol{\nabla}\psi_j\,\mathrm{d}^3\mathbf{x}+\int_{\Omega_b}(\sigma_b\boldsymbol{\nabla}\psi_k)\cdot\boldsymbol{\nabla}\psi_j\,\mathrm{d}^3\mathbf{x},$$

and write it in 2 by 2 block matrix form as

$$\mathcal{K}=\begin{pmatrix}\mathcal{K}_{(1,1)} & \mathcal{K}_{(1,2)}\\ \mathcal{K}_{(2,1)} & \mathcal{K}_{(2,2)}\end{pmatrix}\begin{matrix}\}\text{ size } N\\ \}\text{ size } M\end{matrix}$$

Then, after computing the weak form corresponding to (3.95)–(3.100), it can be

shown that the full finite element linear system to be solved each timestep is:

$$\begin{bmatrix} \frac{\chi\mathcal{C}}{\Delta t}M + K[\sigma_i] & K[\sigma_i] & 0 \\ K[\sigma_i] & \mathcal{K}_{(1,1)} & \mathcal{K}_{(1,2)} \\ 0 & \mathcal{K}_{(2,1)} & \mathcal{K}_{(2,2)} \end{bmatrix} \begin{bmatrix} \mathbf{V}^{m+1} \\ \mathbf{\Phi}_{(1)}^{m+1} \\ \mathbf{\Phi}_{(2)}^{m+1} \end{bmatrix} = \begin{bmatrix} \frac{\chi\mathcal{C}}{\Delta t}M\mathbf{V}^m + \mathbf{c} \\ \mathbf{0} \\ \mathbf{d} \end{bmatrix} \begin{matrix} \} \text{ size } N \\ \} \text{ size } N \\ \} \text{ size } M \end{matrix} \tag{3.103}$$

where the $\mathcal{K}_{(a,b)}$ are sub-blocks of a large $N+M$ by $N+M$ stiffness matrix $\mathcal{K}$, $\mathbf{V} = (V_1, \ldots, V_N)$, $\mathbf{\Phi}_{(1)} = (\Phi_1, \ldots, \Phi_N)$, $\mathbf{\Phi}_{(2)} = (\Phi_{N+1}, \ldots, \Phi_{N+M})$ and $d_j = \int_{\partial\Omega_b \setminus \partial\Omega} I^{(\mathrm{E})} \psi_{j+N} \, \mathrm{d}S$.

In practice, for implementation/parallelization reasons, we introduce a set of dummy voltage values at the bath nodes, $V_{N+1}, \ldots, V_{N+M}$, and add the equations

$$V_j = 0, \qquad j = N+1, \ldots, N+M,$$

so that the linear system becomes of size $(2N + 2M)$.

Letting $\mathbf{V}_{(1)} = \mathbf{V}$, and $\mathbf{V}_{(2)} = (V_{N+1}, \ldots, V_{N+M})$ (i.e. the vector of dummy values), and letting I_M denote the M by M identity matrix, we have

$$\begin{bmatrix} \frac{\chi\mathcal{C}}{\Delta t}M + K[\sigma_i] & 0 & K[\sigma_i] & 0 \\ 0 & I_M & 0 & 0 \\ K[\sigma_i] & 0 & \mathcal{K}_{(1,1)} & \mathcal{K}_{(1,2)} \\ 0 & 0 & \mathcal{K}_{(2,1)} & \mathcal{K}_{(2,2)} \end{bmatrix} \begin{bmatrix} \mathbf{V}_{(1)}^{m+1} \\ \mathbf{V}_{(2)}^{m+1} \\ \mathbf{\Phi}_{(1)}^{m+1} \\ \mathbf{\Phi}_{(2)}^{m+1} \end{bmatrix} = \begin{bmatrix} \frac{\chi\mathcal{C}}{\Delta t}M\mathbf{V}_{(1)}^m + \mathbf{c} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{d} \end{bmatrix} \begin{matrix} \} \text{ size } N \\ \} \text{ size } M \\ \} \text{ size } N \\ \} \text{ size } M \end{matrix} \tag{3.104}$$

3.3.3 The monodomain equation

In cases where the intracellular and extracellular conductivity tensors (σ_i and σ_e) are proportional (i.e. $\sigma_i = \alpha\sigma_e$), the bidomain equations can be reduced to the so-called monodomain equation (see, e.g. [Keener and Sneyd, 1998]). In this model, a PDE in a single unknown, V, is coupled to the system of ODEs describing the

transmembrane ionic currents.

$$\chi\left(\mathcal{C}_m\frac{\partial V}{\partial t}+I_{\text{ion}}(\mathbf{u},V)\right)-\nabla\cdot(\sigma\boldsymbol{\nabla}V) = I^{(\text{vol})}, \tag{3.105}$$

$$\frac{\partial\mathbf{u}}{\partial t} = \mathbf{f}\left(\mathbf{u},V\right), \tag{3.106}$$

where σ is the parallel sum of σ_i and σ_e:

$$\sigma=\frac{\sigma_i\sigma_e}{\sigma_i+\sigma_e} \tag{3.107}$$

and $I^{(\text{vol})}$ a stimulus current with boundary conditions

$$\mathbf{n}\cdot(\sigma\boldsymbol{\nabla}V) = 0, \qquad \text{on } \partial\Omega\,. \tag{3.108}$$

This formulation is sometimes preferred since its solution is computationally less demanding. [Potse et al., 2006] conclude that, in the absence of applied currents, the monodomain model can be used for the study of propagation in the human heart with negligible loss of accuracy. However, it cannot be applied in all situations because it does not permit currents in the extracellular domain to influence $\mathbf{u}$ and V (as occurs when shocks are applied for pacing or defibrillation purposes).

Finite element solution

Using the finite element scheme in Section 3.3.1, the linear system to be solved each timestep can be shown to be

$$\left(\frac{\chi\mathcal{C}}{\Delta t}M+K[\sigma]\right)\mathbf{V}^{m+1}=\left(\frac{\chi\mathcal{C}}{\Delta t}M\mathbf{V}^{m}+\mathbf{c}\right). \tag{3.109}$$

3.3.4 Dealing with linear system singularity

The linear systems (3.93), (3.94), and (3.104) arising from the FEM discretisation of the three forms of bidomain equations presented are singular. This is due to the fact that ϕ_i and ϕ_e only appear in the equations and boundary conditions through their gradient, their solution is therefore defined up to a constant. That is, there are infinitely many solutions but any two solutions differ only by a scalar multiple of $\mathbf{v_1} = (0, \cdots, 0, 1, \cdots, 1)^T$ in (3.93) and (3.104), and a scalar multiple of $\mathbf{v_2} = (1, \cdots, 1)^T$ in (3.94). These vectors form the basis of the kernel of their respective systems.

This situation is common to any elliptic PDE involving Neumann boundary conditions only ($\partial\Omega = \partial\Omega_2$ in (P1)). It is known that if the problem is well-posed (i.e. the compatibility conditions hold) the linear system is always consistent, i.e. the right-hand side is orthogonal to the kernel ($\mathbf{v}_1^T\mathbf{b} = 0$ or $\mathbf{v}_2^T\mathbf{b} = 0$) [Elman et al., 2005]. It can also be shown that $\mathbf{v}_1$ and $\mathbf{v}_2$ are eigenvectors of their respective systems [van der Vorst, 2003]. Both results can be combined to see that if the initial guess $\mathbf{x}^{(0)}$ is a zero vector, then equation (3.81) becomes

$$\mathbf{r}^{(0)} = \mathbf{b} = \sum_j \gamma_j p_0(\lambda_j)\mathbf{z}_j \tag{3.110}$$

and for it to hold, the coefficients γ_j corresponding to the eigenpairs $(0, \mathbf{v}_1)$, or $(0, \mathbf{v}_2)$ must be zero (since the right-hand side $\mathbf{b}$ does not have any component in the direction of $\mathbf{v}_1$ and $\mathbf{v}_2$, i.e. it is orthogonal to them). Hence, we can conclude that the fact that, by definition, $p_i(0) = 1$ is not a problem provided that the system has a consistent right-hand side, and therefore CG and CI can be applied to the symmetric positive semi-definite linear systems considered in this Thesis.

3.4 Chaste

In the previous section, we presented the semi-implicit FEM discretisation of the bidomain equations that we will use in this work. In the current section, we introduce the software platform where both this algorithm and the rest of solutions presented in this Thesis have been implemented: Chaste.

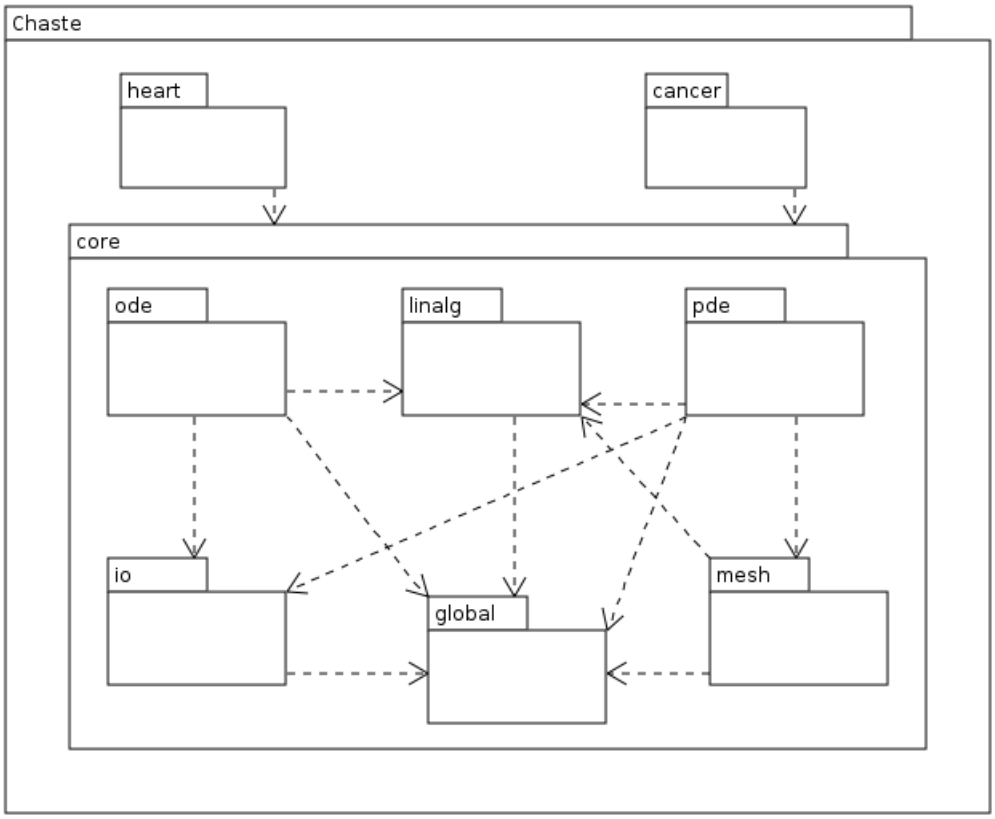
Chaste [Pitt-Francis et al., 2009] is an open source computational framework for the simulation of systems in biology. Chaste is distributed under the LGPL license and can be downloaded from `www.cs.ox.ac.uk/chaste`. Chaste aims to be extensible, robust, fast, accurate and maintainable and to use state-of-the-art numerical techniques. Chaste tries to fill the gap of the lack of open source, fast and well-engineered simulation software for Systems Biology problems. Appendix A provides a description of the software engineering methodology used to ensure robustness and maintainability.

Chaste has been designed with the aim of being a multi-purpose library supporting computational simulations for a wide range of biological problems. While it is a generic extensible library, software development to date has focused on the relatively mature area of cardiac electrophysiology and the less well-developed areas of tissue growth and cancer growth.

Figure 3.5 shows a schematic of the modular structure within Chaste. The mathematical modelling of both cardiac and cancer domains relies on (i) accurate representation of a complex, realistic geometry provided primarily through unstructured meshes; (ii) temporal and spatial discretisation of PDEs, mainly with the finite element method; (iii) solution of large sparse systems of linear (or non-linear) equations and (iv) the numerical solution of ODE systems. These four common features of the code become four component libraries: `mesh`, `pde`, `linalg` and `ode` respectively. They are supported by two lower-level libraries, `global` (which is responsible for initialisation and book-keeping) and `io` (which handles input and output). Built

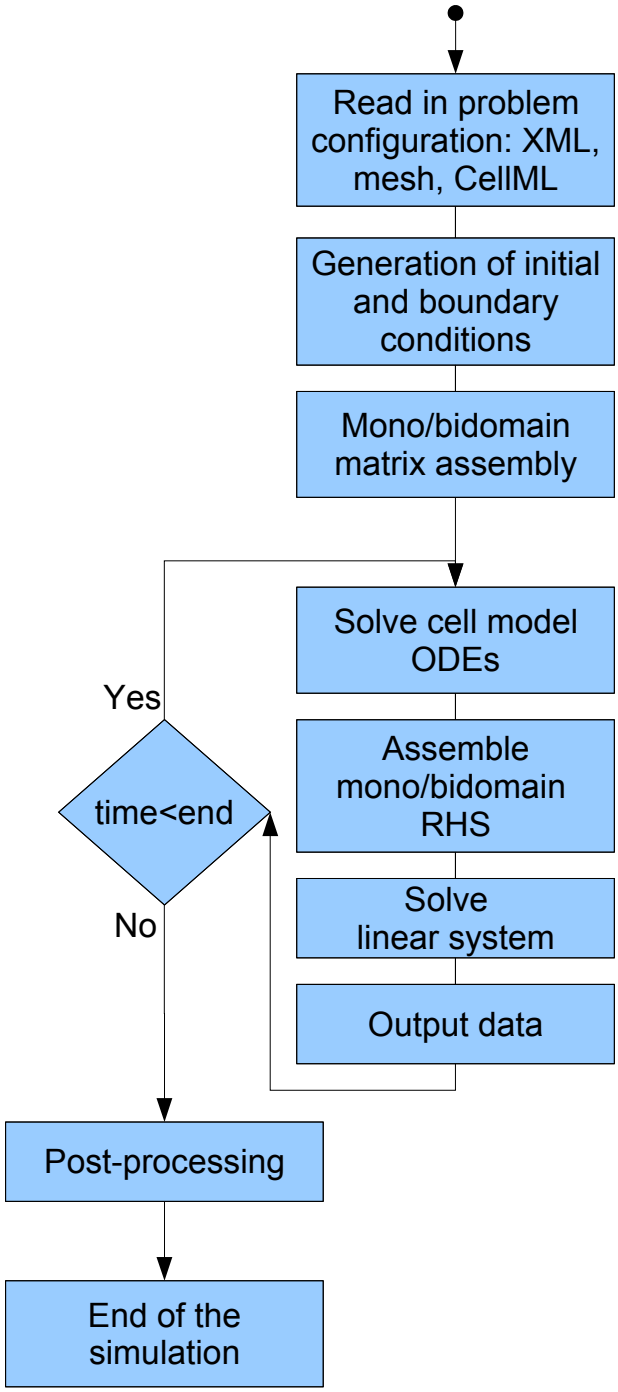
upon these are the two modelling components of Chaste, `heart` and `cancer`

Figure 3.5 The structure of Chaste's code-base. Arrows represent dependencies between components. The 'core' component is merely a convenient shorthand for referring to the libraries shown within it. Both the heart and cancer libraries depend on all core libraries. Image originally from [Pitt-Francis et al., 2009].



Code for the `ode` component is automatically generated from models found in the CellML repository with `PyCml`. `PyCml`[3] is a project developing CellML processing tools using the Python programming language. `PyCml` provides validation and automatic generation of highly-optimised C++ code for CellML models [Cooper et al., 2006].

Figure 3.6 A schematic of the Chaste's cardiac electrophysiology simulator main components.



3.4.1 Main components of Chaste's cardiac electrophysiology simulator

In order to facilitate discussion in later chapters, we describe here the main stages of Chaste's cardiac electrophysiology simulator. Figure 3.6 presents them graphically, highlighting the iterative nature of the solution algorithm. At the beginning of the simulation, the problem description is read from file, which involves: reading in an XML configuration file describing the simulation, loading the geometrical model from disk (i.e. the mesh), and (optionally) reading and pre-processing a set of CellML files describing ionic kinetics. After this, and based on the original problem description, a set of initial and boundary conditions are generated. In the case of cardiac electrophysiological simulations, where the FEM system matrix is constant in time, the mono/bidomain system matrix is assembled before moving into the iterative part of the simulation. These three stages constitute the initialisation part of the simulation.

After this, the iterative part of the simulation starts. Based on the simulation duration and the PDE timestep specified, a number solution steps are performed iteratively. Each of these steps involves: integration of the individual cell models at each mesh node (ODE solution), assembly of the right-hand side vector in (3.104) or (3.109), solution of the mono/bidomain linear system, and, finally writing to disk the solution vector or a subset of it.

Once the last timestep has been solved, an optional post-processing stage may happen. It includes computation of relevant electrophysiological values (e.g. action potential duration, activation maps, upstroke velocity) and/or conversion from Chaste's standard output format to others required by specific visualisation/post-processing tools.

[3] `https://chaste.cs.ox.ac.uk/cellml/`

Chapter 4

Towards real-time simulation of cardiac electrophysiology: parallel scalability improvements

In silico studies of shock-induced arrhythmogenesis in the human heart are currently not feasible due to limitations in the simulation technology available. These limitations stem from the need of solving the bidomain equations with precise description of fibre structure on spatially very accurate human geometries and with detailed representation of membrane kinetics. We saw in the previous chapter how the computational complexity of the FEM-based solution of the bidomain equations is a function of the number of degrees of freedom (DOFS) in the geometrical model of interest. The increase in model size required for the study of shock-induced arrhythmogenesis in the human heart directly translates into a considerable increase in computational requirements when compared to similar studies in small mammalian geometries.

We have seen in recent years a number of technological breakthroughs which have led to a remarkable increase in computational resources available to simulation scientists (e.g. many-core architectures, petascale computing). Unfortunately, the

approach to improving CPU performance has changed due to technological barriers. Instead of increasing clock frequencies in order to achieve higher FLOPS[1] rate, CPU manufacturers are delivering higher performances by integrating more CPU cores[2] together. This shift has invalidated some of the approaches to scientific simulation, e.g. sequential simulation environments.

Therefore, the big challenge that simulation software is currently facing is the need to run simulations with models of a complexity never seen before on a new generation of massively parallel architectures in order to be able to answer some of the open questions in cardiac science.

4.1 Introduction

There exists a large body of literature concerning the development of parallel cardiac electrophysiology solvers (see [Bordas et al., 2009; Clayton et al., 2011; Linge et al., 2009] for surveys). Examples of early contributions to parallel solution approaches are [Cai and Lines, 2002; Fishler and Thakor, 1991; Ng et al., 1995; Pollard and Barr, 1991; Quan et al., 1998; Saleheen et al., 1997; Winslow et al., 1993]. Common amongst most of these works is that they considered explicit solution schemes for the monodomain model, which can be parallelised very efficiently, and that they used shared-memory architectures[3]. An early contribution to distributed-memory[4]

[1] Floating point operations per second. A measure of how many arithmetical operations a given architecture can perform per second.

[2] In this work we will consider the terms core and processor to be synonymous.

[3] A shared-memory architecture is one where two or more processors are connected to a single shared main memory and are controlled by a single operating system instance. Any processor can access any memory position, although in some designs different latencies may apply. One of the advantages of shared-memory architectures is programming ease. One of its main drawbacks is that the design does not scale to as many processors as the distributed-memory counterpart.

[4] A distributed-memory architecture is one where two or more processors equiped with local memory are connected through a communication network or bus. Processors can only directly access its own local memory and explicit message passing has to be programmed in order to gain access to data located at any other processor's memory. The main advantage of this approach is that the design scales to a potentially unlimited number of processors (over half million has been reported at `www.top500.org`). Developing code that uses this kind of massively parallel architectures is, however, very challenging.

approaches was made by [Porras et al., 2000], who compared four different parallel schemes for the two-dimensional monodomain equations. More recently, [Vigmond et al., 2002] present parallel computations on shared-memory architectures with two to four processors and compared the performance of a number of direct and iterative linear solvers.

In the context of distributed-memory architectures there exist a number of works that use the MPI standard[5] and the PETSc library (see Section 3.2.4 for a description of the library) for the development of parallel bidomain simulators. Early examples are [Colli-Franzone and Pavarino, 2004; Vigmond et al., 2003]. A similar PETSc-based approach is used in [dos Santos et al., 2004] combined with parallel geometric multigrid preconditioning (see Section 3.2.3 for a description of this method). [Murillo and Cai, 2004] also use the PETSc library to devise a parallel bidomain solver based on a fully implicit time discretisation. More recently, [Plank et al., 2007] have extended the solver presented in [dos Santos et al., 2004] to use algebraic multigrid preconditioning (again, see Section 3.2.3 for a description of this method). The same solver was adapted for the solution of the monodomain equation in [Niederer et al., 2011a] achieving close to optimal scalability with up to 1024 cores with both explicit and semi-implicit schemes and around 40% efficiency with an explicit scheme and 16,384 cores. With the same number of cores, [Reumann et al., 2009] achieve a 71% efficiency using the explicit finite difference method for the solution of the monodomain equation. Finally, [Vázquez et al., 2011] adapt a large-scale computational mechanics simulation platform to the solution of the monodomain equations with an explicit scheme achieving almost linear scaling for up to 1000 processors.

A performance comparison of the aforementioned solvers is very difficult. As we saw, they use different numerical schemes, they are evaluated on platforms with

[5]Message passing interface (MPI) is a standard for distributed memory programming. It provides the basic infrastructure for message passing, synchronisation and collective operations required for distributed memory parallel programming.

major architectural differences and authors do not usually release enough information about solution timesteps and tolerances so that the level of accuracy of their solutions can be compared[6]. A performance metric often reported by authors is the ratio between the time taken to perform a simulation and the amount of time simulated (i.e. real-time ratio). Table 4.1 summarises some of the values found in the literature.

Table 4.1 Ratio between time taken to perform a simulation and amount of time simulated. (*) [Vázquez et al., 2011] report number of tetrahedra in the mesh: 17 million, but not number of degrees of freedom.

Reference	# of cores	# of DOFS	Real-time ratio
[Potse et al., 2006]	32	26 million	111,000
[ten Tusscher et al., 2007]	20	13.5 million	43,200
[Reumann et al., 2009]	16,384	32.5 million	13,180
[Vázquez et al., 2011]	500	(*)	450
[Niederer et al., 2011a]	16,384	26 million	240

It can be appreciated how the solvers reported to run efficiently in hundreds of processors or more (i.e. the last three in Table 4.1) share one or more of the following four limitations: i) they use the monodomain model, ii) they use explicit time discretisations, iii) they use spatial discretisation methods that only allow the use of regular grids (e.g. finite difference method), and/or iv) they are not freely available to the scientific community. In this Thesis we are interested in developing an open source cardiac simulation technology that achieves a similar degree of performance in large scale HPC infrastructures using the bidomain equations and a semi-implicit time discretisation (Section 3.3 describes the method). The reasons are three-fold:

- Our application of interest, human shock-induced arrhythmogenesis (see Chapter 2 for a description), requires the use of the bidomain equations, since the monodomain model can not properly model defibrillation processes.

[6]The use of iterative numerical methods in some of the implementations leads to the situation where one can relax tolerances in order to reduce computational cost.

- Explicit time discretisation imposes a constraint on the maximum timestep directly proportional to the grid edge length (see Section 3.1.3 for more details). In this scenario, the increase in the level of detail of the geometrical models described at the beginning of the chapter will inevitably lead to increasingly shorter timesteps. This situation is likely to have a negative impact on overall performance. We want to consider unconditionally stable methods only (e.g. semi-implicit time discretisation, see Section 3.3 for a description of this method).

- Unstructured grids allow for more realistic representation of ventricular surfaces and fine-grained features than structured grids. The finite element method is preferred for the spatial discretisation of the bidomain equations for its support of unstructured grids (again, see Section 3.3 for a description of this method).

- We believe that open sourcing our simulation technology is: i) the only way to facilitate the reproducibility of our results, and ii) a contribution towards the shared scientific endeavour.

In this chapter, we will describe how we have tackled the problem of transforming Chaste into a state-of-the-art parallel computing platform for cardiac electrophysiology research. We will use as a case study the experience of getting Chaste to run efficiently with the Oxford rabbit model (see Section 2.3 for a description of the model) in up to 4096 cores. At the begining of the project, Chaste was not capable of running simulations with this model due to the excessive memory footprint of the data structures representing the mesh. The next section outlines how the problem was tackled.

4.2 Chapter structure

The rest of the chapter is structured as follows. In Section 4.3, we describe the improvements implemented in Chaste in order to run simulations with the Oxford rabbit model in small to medium size multicore systems and clusters. This task mainly involved solving the problem of efficiently partitioning and distributing the large computational mesh (3.7M nodes) using graph-partitioning algorithms such as those in the METIS library. Before the Oxford rabbit model was available, the geometrical model with the highest level of detail that we had access to was the UCSD rabbit model, with 64K nodes. The two orders of magnitude increase in the number of nodes translates into a two orders of magnitude increase in the memory footprint of the data structure representing the model. In Chaste's original implementation, the whole data structure was replicated by all the processes involved in a simulation. Without domain decomposition, the memory footprint of the Oxford rabbit model was often larger than the available physical memory. Note that this was never a problem with the UCSD rabbit model. Following these improvements, Chaste was capable of running bidomain simulations with state-of-the-art geometrical models. However, parallel speedup[7] used to saturate at a value around 120 due to excessive communications and portions of the solver running sequentially. Publications [Bernabeu et al., 2008, 2009] were possible, thanks to this improvement.

Section 4.4 presents the experience of porting Chaste to a large-scale High Performance Computing (HPC) infrastructure such as HECToR (UK's high-end computational resource) as part of the effort of making Chaste petascale-enabled. This includes a comprehensive analysis of the scalability of all of the stages involved in a cardiac electrophysiology simulation, identifying scalability bottlenecks and addressing them with the design and implementation of novel computational and numerical

[7]Parallel speedup is a measure of how good a piece of code is at utilising parallel hardware. It is defined as $S_p = \frac{t_1}{t_p}$, where t_1 is the time require to run a given simulation sequentially and t_p the time taken by the same simulation on p processors. Ideal (or linear) speedup is $S_p = p$

techniques. Upon completion of this task speedups of over 1400 and real-time ratios of 210 with 7.4M degrees of freedom and 4096 processors were achieved. These results are comparable to those in Table 4.1 and overcome the limitation mentioned before: they correspond to bidomain simulations with unconditionally stable numerical methods on unstructured grids and the code is freely available. In the case of monodomain simulations, the ratio goes down to 45 with 3.7M DOFS and 4224 processors and 2.8 with 64K DOFS and 96 processors. These improvements have been essential in order to achieve the level of performance necessary to run the simulation study presented in Chapter 7.

Section 4.4 was developed in the framework of the multi-institutional project preDiCT and it is, therefore, a collaborative effort. Work presented here is original by the author unless otherwise credited.

4.3 Implementing domain decomposition

In our first simulation with the Oxford rabbit model we used an 8-core Xeon server with 16GB of RAM. However, it was only possible to use one of the cores and the whole main memory of the system. The reason was that, despite Chaste being a fully parallelised solver, the data structure representing the mesh in memory was being replicated at each of the MPI processes involved in the simulation[8]. Using more cores would have implied replicating the whole data structure multiple times. This would have quickly exhausted the system resources. Note that this had never been a problem with any of the cardiac models used to the date (e.g. UCSD model) due to the smaller number of nodes.

Domain decomposition was already happening in other components of the simulator. For instance, the systems of ODEs describing the reactive part of the bidomain

[8]In a pure distributed memory software like Chaste, one MPI process has to be launched to utilise each of the system cores.

equations are not replicated in all of the processors running a simulation. A simulation on a mesh with n nodes requires, generally, n ODE solver objects[9] to be instantiated. In a simulation with p processors, each processor will instantiate, approximately, $\frac{n}{p}$ objects. In the same way, the system of linear equations arising from the FEM discretisation is partitioned row-wise and distributed among the available processors (see Section 3.2.4 for a description).

An important design decision concerns the way that the computational domain (i.e. the mesh) is partitioned. In principle, one can choose to do it node-wise or element-wise. This choice has important consequences in terms of load balancing. In this section we will see how the linear system solution and the ODE solution stages dominate total execution time for simulations with realistic 3D geometries. In both cases, the execution time at each subdomain is a function of the number of grid points assigned to the subdomain (i.e. number of degrees of freedom in the linear system and number of cell models to be integrated, respectively). Therefore, it makes sense to partition node-wise to ensure that the total number of grid points is as evenly distributed as possible among the available subdomains. Partitioning element-wise would instead optimise the element distribution (which would actually have a positive impact on stages like matrix assembly, for instance). Unfortunately, it would potentially generate a suboptimal node distribution and therefore unbalance the stages that dominate total execution time.

In this section we will describe an algorithm for partitioning and distributing the mesh data structure node-wise among the processors involved in a simulation. More formally, let $M = (V, T)$ be a computational mesh. $V = \{v_1, \ldots, v_{n_v}\}$ is the set of vertices of the mesh and $T = \{t_1, \ldots, t_{n_t}\}$ is the set of elements such that $t_i \subset V$. An equivalent graph representation of the mesh, $G = (V, E)$, can be constructed such that $E = \{e_1, \ldots, e_{n_e}\}$ is the set of edges in the mesh and

[9] In object-oriented programming jargon, an object is a runtime instance of a class. A class is a programming abstraction that encapsulates data and operations relevant to a given entity together.

$e_i = (v_j, v_k) \in (V \times V) : v_j, v_k \in t_l \wedge j \neq k$.

The problem of partitioning a mesh can then be recast as the k-way graph partitioning problem [Karypis and Kumar, 1998]: given a graph $G = (V, E)$, partition V into k disjoint subsets, $V_1, V_2, \ldots, V_k : V = \bigcup V_i$, such that: i) $|V_i| = \frac{n_v}{k}$, and ii) the number of edges of E whose incident vertices belong to different subsets is minimised.

Algorithms that find good partitions of highly unstructured graphs are critical for the development of efficient solutions for a wide range of applications on both serial and parallel computers. In our case, the goal of the first condition is to balance the computations among the processors. The goal of the second condition is to minimize communication among processors. Graph partitioning can be used to successfully satisfy both conditions if the finite element mesh is first described as a graph and then partitioned using well-stablished algorithms.

4.3.1 Assigning node ownership: graph partitioning and METIS

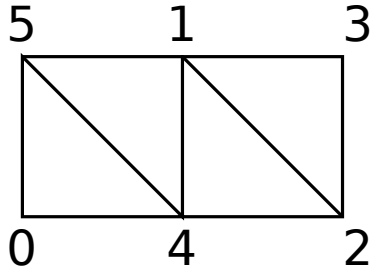
METIS[10] is a library that implements state-of-the-art algorithms for graph partitioning. It defines a format for representing graphs, an API for handling them, and several partitioning techniques. We can pass to METIS a graph-like representation of our mesh and it will return a partition with the characteristics described above. To illustrate how METIS performs a graph partition, we consider the simple example mesh illustrated in Figure 4.1.

For this mesh, an optimal two-processor (2-way) partition is:

$$V_0 = \{0, 4, 5\}, V_1 = \{1, 2, 3\} \tag{4.1}$$

[10] `http://www.cs.umn.edu/~karypis/metis`

Figure 4.1 Simple mesh to be distributed between two processors.



This partition cuts three edges which is the minimum that can be achieved if the condition $|V_0| = |V_1|$ is enforced.

Unfortunately, this partition is not compatible with the parallel layout of PETSc matrices (see Section 3.2.4 for a description). Only consecutive nodes can be assigned to a processor; hence there is no way of assigning rows 0, 4, and 5 to the first processor. A way around this problem is to use the previous partition to define a renumbering of the nodes that ensures that the contiguity condition holds. The new index for node i is given by the following expression:

$$R(i) = l + \sum_{k=0}^{j-1} |V_k|, \quad i = V_j[l] \tag{4.2}$$

In our model problem we have:

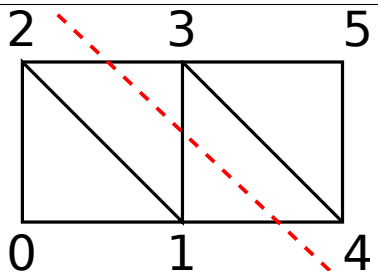
i	0	1	2	3	4	5
$R(i)$	0	3	4	5	1	2

If we apply the previous permutation to (4.1), it becomes PETSc compatible as shown in Figure 4.2.

4.3.2 Assigning element ownership

In the previous subsection we described how to generate a partition, $\{V_1, \ldots, V_k\}$, of the set of vertices V of our mesh $M = (V, T)$. We will now show how to generate a partition of the set of elements T based on the vertex partition.

Figure 4.2 Simple mesh with nodes renumbered. Dashed line shows parallel partition.



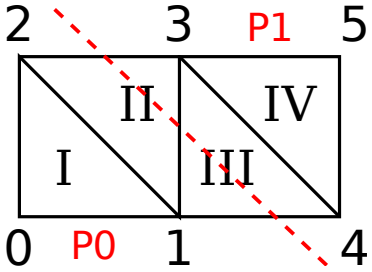
We saw in Section 3.1.6 that in order to assemble the system matrix, A, we loop over the elements in T, compute its local contribution $E_{t_i \in T}$, and map that contribution in the global matrix (using a boolean rectangular matrix $L_{t_i \in T}$ defining the global index of each of the vertices in the element),

$$A = \sum_{t_i \in T} L_{t_i}^T E_{t_i} L_{t_i} \,. \tag{4.3}$$

We saw in Section 3.2.4 that A is partitioned row-wise based on the vertices assigned to each processor i, V_i. In order to avoid communication when assembling the matrix, we ensure that a given element is assembled by the processor owning the matrix rows corresponding to the vertices in the element. Based on this idea, one can easily assign most of the elements in T. However, for the elements located at the border between two partitions (such as II and III in Figure 4.3) one has to design a strategy for ownership assignment. Some options are: i) assigning each of them to the processor owning the largest number of nodes from the element, ii) allowing element multiple ownership or iii) distributing them in a way that the number of elements owned by each processor is balanced.

Depending on the strategy chosen, different patterns of communication and load balance properties will emerge when assembling the linear system. Choosing strategies i) or iii) leads to a situation where contributions to certain rows are generated by processors different from the row owner and therefore data need to be communi-

Figure 4.3 Model problem geometry. Dashed line shows parallel partition.



cated. In contrast, strategy ii) ensures no data communication when assembling the system matrix at the cost of replicating some computation. This approach may not seem reasonable in Figure 4.3 since it implies that both processors assemble 50% more elements. However, in realistic geometrical models $|T| \gg p$ and the number of elements at the border between processors is therefore just a small fraction of the total. Furthermore, the volume of computation replicated is proportional to the number of edges cut by the node partition. The algorithm for node partitioning presented in Section 4.3.1 minimises the number of edges cut and therefore minimises the number of elements assembled by multiple processors.

More formally, let T be the set of mesh elements partitioned into p potentially overlapping subdomains T_i such that:

$$T = \bigcup_{i=0}^{p-1} T_i \tag{4.4}$$

and consider a row-based distribution of the system matrix

$$A = \begin{bmatrix} A_0 \\ \vdots \\ A_{p-1} \end{bmatrix} \tag{4.5}$$

with $A_i \in \mathbb{R}^{\frac{n}{p} \times n}$ the rectangular block of A owned by processor i (assuming $n = kp$,

$k \in \mathbb{N}$ for simplicity). Let $M_{e,i} = [m^i_{jk}] \in \mathbb{R}^{n_e \times \frac{n}{p}}$,

$$m^i_{jk} := \begin{cases} 1, & \text{if } l_{j((i-1)\frac{n}{p}+k)} = 1 \\ 0, & \text{otherwise} \end{cases}, \tag{4.6}$$

be the boolean matrix that maps E_e into A_i. See Section 3.1.6 for the definition of l_{jk}. It can be shown that A_i can be assembled from local data without need of communications, i.e.

$$A_i = \sum_{t_i \in T_i} M^T_{e,t_i} E_{t_i} M_{e,t_i}\,, \tag{4.7}$$

only if strategy ii) is used.

4.3.3 Performance impact

The first obvious benefit of the domain decomposition algorithm described in this section is that after its implementation Chaste was capable of running parallel simulations with the Oxford rabbit model, which was not possible before. However, the method presented also speeds up simulations with smaller models regardless of the mesh data structure being replicated or not. This is due to the fact that METIS computes a problem partition that minimises computation replication in the matrix assembly stage and communication between processors in other parts of the simulator (such as system solution).

In order to quantify the impact we designed a benchmark consisting of 10 ms of bidomain cardiac activity in the Oxford rabbit model with a Luo Rudy 1991 ionic model [Luo and Rudy, 1991] and ran it with the new domain decomposition algorithm with and without METIS for computing the node partition. In the non-METIS version we used the original node numbering to generate $\{V_1, V_2, \ldots, V_k\}$. Figure 4.4 compares both approaches in the UCSD model for a simulation with 4 processors. It can be appreciated how nodes assigned to each processor are arranged

together and partition borders are reduced.

As we explained in Section 4.3.2, the minimisation in the number of edges going across multiple partitions also translates into fewer mesh elements being replicated. Table 4.2 compares the number of elements assigned to each processor in both vertex partitioning approaches for a simulation with the UCSD model and 16 processors. The average number of elements per processor in the METIS case is 6.78%, close to the theoretical optimum of 6.25.

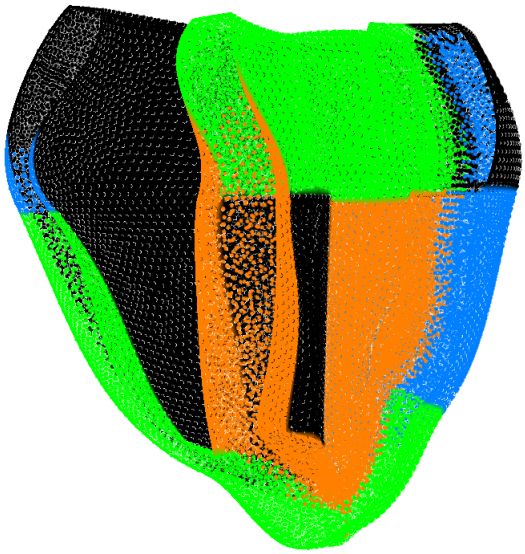
Table 4.2 Comparison between the two partition schemes presented.

	with METIS		without METIS	
processor number	# local elements	%	# local elements	%
0	23524	7.3	20979	6.51
1	21775	6.76	25264	7.84
2	22859	7.09	28688	8.9
3	21374	6.63	30469	9.45
4	21492	6.67	31428	9.75
5	20993	6.51	20495	6.36
6	23045	7.15	30539	9.48
7	23309	7.23	28467	8.83
8	19856	6.16	23473	7.28
9	22534	6.99	25634	7.95
10	19013	5.9	27494	8.53
11	19830	6.15	31889	9.9
12	22707	7.05	28363	8.8
13	22303	6.92	31873	9.89
14	23027	7.15	31650	9.82
15	22211	6.89	30523	9.47
average	22562	6.78	30602	8.67

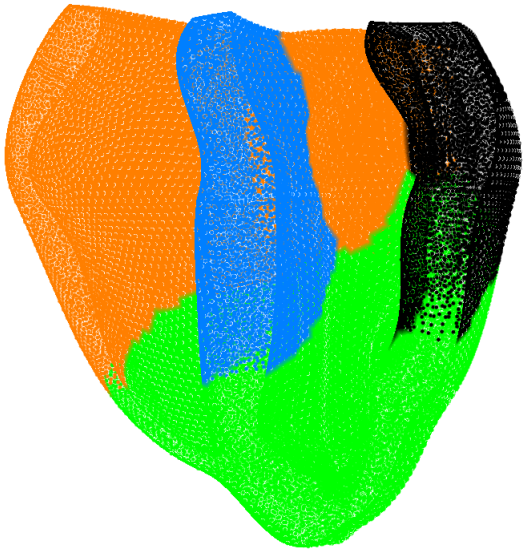
Finally, Tables 4.3 and 4.4 present a breakdown of the time required to run the benchmark with multiple processor configurations. Table 4.5 shows the speedup achieved in each of these stages. Figure 4.5 plots the speedup of the overall simulation.

As expected, using METIS to compute the vertex partitions speeds up the assembly of the system matrix and the right-hand side by a factor of 91% and 65% for 32 processors. This is due to processors owning fewer duplicated elements and

Figure 4.4 UCSD model partitioned into 4 sub-domains. In panel (a) with a naive strategy based on the original numbering. In panel (b) with the algorithm presented in this section. Nodes owned by a given processor are displayed with the same colour.



(a)



(b)

Table 4.3 Execution time breakdown (in seconds) for simulations with the original partitioning scheme.

Number of processors	Matrix assembly	ODE solution	RHS assembly	System solution	Other	Total
1	1350	6740	1260	30900	550	40800
2	632	3620	685	27300	563	32800
4	358	1880	396	14000	366	17000
8	166	954	239	7290	301	8950
16	78.1	482	124	3700	275	4660
32	38.9	242	65.8	1890	273	2510
64	19.3	120	39	1090	281	1550

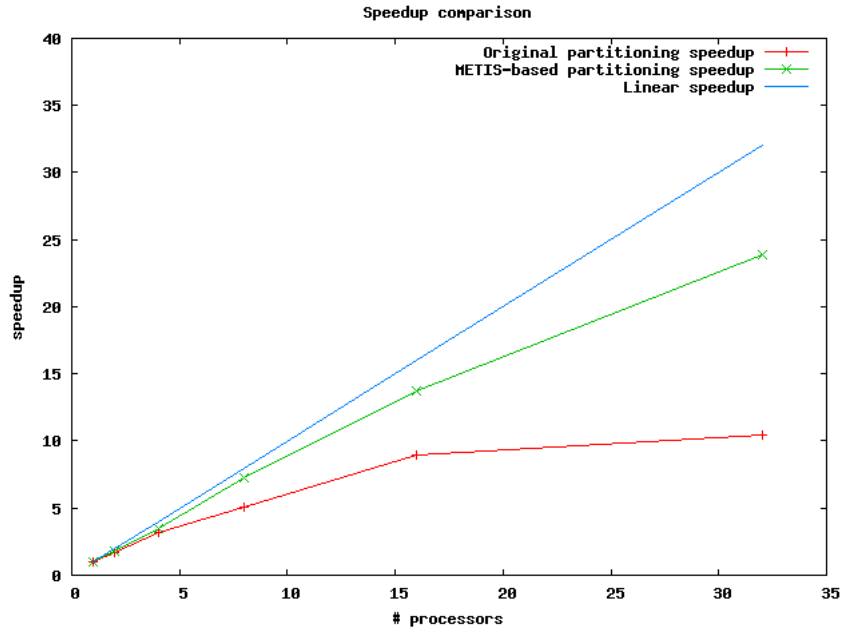
Table 4.4 Execution time breakdown (in seconds) for simulations with METIS partitioning scheme.

Number of processors	Matrix assembly	ODE solution	RHS assembly	System solution	Other	Total
1	1350	6740	1260	30900	550	40800
2	526	3620	629	15400	725	20900
4	206	1880	317	7920	477	10800
8	86.9	954	155	4370	384	5950
16	43.2	481	78.2	2150	377	3130
32	20.4	242	40	1050	377	1730
64	10.9	120	24.5	654	390	1200

Table 4.5 METIS vs original partition speedup. Breakdown and totals.

# of processors	Matrix assembly	ODEs	RHS assembly	System solution	Total
2	1.20	1	1.09	1.77	1.57
4	1.74	1	1.25	1.77	1.57
8	1.91	1	1.54	1.67	1.50
16	1.81	1	1.59	1.72	1.49
32	1.91	1	1.65	1.80	1.45
64	1.77	1	1.59	1.67	1.29

Figure 4.5 Parallel perfomance comparison.



therefore minimising the number of redundant operations. A speedup of 80% is reported in the system solution stage for the same number of processors. This improvement was not originally expected and the next subsection analyses it in more detail. Overall, simulations are sped up by 45%–57% for $p \leq 32$. For $p = 64$, a 77%, 59%, and 67% speedup in the matrix assembly, RHS assembly, and linear system solution stages only translates into a 29% global speedup. This is due to scalability problems in stages not covered in this breakdown which will be investigated in Section 4.4. Before moving on to these issues, we first turn our attention to the linear solver scalability implications of the partitioning approach described before.

Communication reduction in the linear system solution

The system of linear equations arising from the FEM discretisation of the bidomain equations is often solved with the CG algorithm (see Section 3.2 for details). The main computational kernel in CG is the matrix-vector product in line 10 of

Algorithm 1:

$$\mathbf{q} = A\mathbf{p} \tag{4.8}$$

with $A \in \mathbb{R}^{n \times n}$ and $\mathbf{q}, \mathbf{p} \in \mathbb{R}^n$. In parallel, this is a tightly-coupled operation since, in non-trivial cases, no process owns enough data to perform its operations independently from the rest, requiring a certain degree of communication. Assuming the row-based distribution of matrices and vectors described in Section 3.2.4, (4.8) can be rewritten as:

$$\begin{pmatrix} \mathbf{q}_0 \\ \mathbf{q}_1 \\ \vdots \\ \mathbf{q}_{p-1} \end{pmatrix} = \begin{pmatrix} A_0 \\ A_1 \\ \vdots \\ A_{p-1} \end{pmatrix} \begin{pmatrix} \mathbf{p}_0 \\ \mathbf{p}_1 \\ \vdots \\ \mathbf{p}_{p-1} \end{pmatrix}$$

with $A_i \in \mathbb{R}^{\frac{n}{p} \times n}$ and $\mathbf{q}_i, \mathbf{p}_i \in \mathbb{R}^{\frac{n}{p}}$ (assuming $n = kp$, $k \in \mathbb{N}$ for simplicity). Processor i owns sub-blocks A_i and $\mathbf{p}_i$ and is responsible of computing $\mathbf{q}_i$:

$$\mathbf{q}_i = A_i \begin{pmatrix} \mathbf{p}_0 \\ \mathbf{p}_1 \\ \vdots \\ \mathbf{p}_{p-1} \end{pmatrix} . \tag{4.9}$$

It can be observed that, in non-trivial cases, processor i needs non-local sub-blocks of $\mathbf{p}$ in order to compute $\mathbf{q}_i$. In order to understand which parts of $\mathbf{p}$ need to be

communicated, let us rewrite (4.9) as

$$\mathbf{q}_i = \begin{pmatrix} \mathcal{A}_0 & \cdots & \mathcal{A}_i & \cdots & \mathcal{A}_{p-1} \end{pmatrix} \begin{pmatrix} \mathbf{p}_0 \\ \vdots \\ \mathbf{p}_i \\ \vdots \\ \mathbf{p}_{p-1} \end{pmatrix}$$

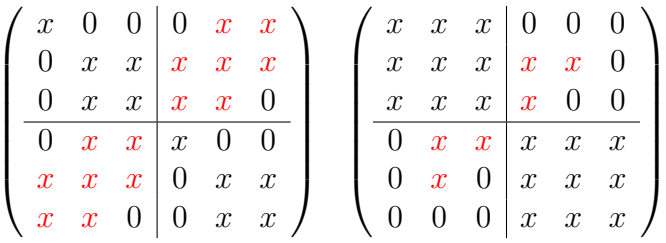
with $\mathcal{A}_j \in \mathbb{R}^{\frac{n}{p} \times \frac{n}{p}}$. Sub-block $\mathcal{A}_i$ corresponds to the block sitting in the diagonal of A, while $\mathcal{A}_j, i \neq j$ are the, so-called, off-diagonal sub-blocks. Communications are required to compute $\mathcal{A}_j \mathbf{p}_j, i \neq j$ since $\mathbf{p}_j, i \neq j$ is not local. $\mathcal{A}_i \mathbf{p}_i$ is communication free.

Since $\mathcal{A}_j$ is potentially sparse, not all the entries in $\mathbf{p}_j, i \neq j$ need to be sent to processor i. Entry l of $\mathbf{p}_j, j \neq i$ is required by processor i if and only if $\exists k : \mathcal{A}_j(k, l) \neq 0$. Therefore, minimising the non-zero entries in the off-diagonal sub-blocks of A minimises the amount of communication required to perform a parallel matrix-vector product.

Figure 4.6 shows the non-zero structure of the monodomain matrices assembled from both partition examples compared throughout this section (Figures 4.1 and 4.2). It can be seen that minimising the communication border between processors minimises the number of non-zero entries in the off-diagonal blocks of the matrix.

Another way of looking into this result is to see the system matrix as the connectivity matrix of the graph associated with the computational mesh. A matrix coefficient $a_{ij} \neq 0$ represents an edge defined between nodes i and j. Non-zero entries in the diagonal block represent edges defined between local nodes. Non-zero entries in the off-diagonal block represent edges defined across two different partitions. Therefore, minimising the communication border between processors minimises the communications required in the matrix-vector product and improves

Figure 4.6 Non-zero structure comparison. Communications are proportional to the number of off-diagonal non-zero entries (in red).



naive partition METIS partition

the scalability of the CG algorithm as seen in Table 4.5. In our model problem, communications are reduced by 33%.

Having considered the problem of efficiently partitioning large computational meshes using graph-partitioning techniques in order to enable bidomain simulations with state-of-the-art cardiac models in Chaste, we now turn our attention to the problem of scaling our bidomain solver to high-end computational resources.

4.4 Scaling Chaste to high-end computational resources

With the arrival of petascale computing and the advent of the exascale era, we have seen a remarkable increase in computational power available to scientists. This exciting technological advance has paved the way to more and more complex simulation studies in many fields of science. The consequences of this shift are twofold. Firstly, an increase in computational resources never seen before in the form of more CPU cores, larger amounts of memory, and faster interconnection technology. Away from the idea of a single execution thread that computing was born with, our programs need to be aware of millions of independent execution units that need to be effi-

ciently coordinated towards a common goal. Secondly, a progressive increase in the volume of input data that our algorithms need to handle must also be considered.

Petascale hardware is becoming more accessible to the scientific community with a total of 10 machines — as of June 2011 — achieving a sustained LINPACK[11] performance of 1 petaFLOPS[12] or more, worldwide[13]. A reduced group of high-level scientific codes have also broken the petaflops barrier[14] (e.g. physics, materials science, and chemistry). To the best of our understanding, no code using FEM for the solution of cardiac electrophysiology problems has achieved this milestone. As a first step in that direction we will describe the experience of porting Chaste to a cutting-edge HPC resource such as HECToR (UK's high-end computational resource and 24th fastest machine in the world[15]).

In this section, we propose effective parallelisation strategies for all the stages of a mono/bidomain simulation presented in Section 3.4.1, highlighting where parallel bottlenecks remain. The parallelisation of each of these stages (including all the algorithms and data structures involved) cannot be seen as independent tasks. The effective parallelisation of the whole cardiac electrophysiology simulator requires considering how design decisions taken for some portion of the solver will affect other parts. Furthermore, the recent increase in number of cores available makes certain operations often not considered a major threat to parallel scaling become potential bottlenecks (e.g. synchronisation, I/O).

When solving systems of coupled PDEs (such as the bidomain equations) with FEM, several fields are computed at each discretisation point (two in our case). In this context, a design decision regarding how unknowns are arranged in the linear

[11] The LINPACK benchmark is used to measure the floating point computing power of a given system. It measures how fast a computer solves a dense $n \times n$ system of linear equations $A\mathbf{x} = \mathbf{b}$. The solution is obtained by Gaussian elimination with partial pivoting. The result is given in FLOPS.

[12] 10^{15} FLOPS

[13] According to `www.top500.org`

[14] `www.olcf.ornl.gov/2010/11/15/ornl-systems-lead-in-petascale-science/`

[15] As of June 2011.

system has to be taken. The options are to use either a 'blocked' distribution, an 'interleaved' distribution or some hybrid approach. In a sequential algorithm, the choice has a moderate impact in performance often associated with matrix bandwidth reduction. In parallel, there exists an important correlation between the approach taken and the volume of communications required for the assembly and solution of FEM linear systems. We discuss some of the implications later in this section.

In Chaste's bidomain solver, we decided to use an interleaved approach where the unknowns in linear system (3.104) are ordered $[V_1^m, \Phi_1^m, V_2^m, \Phi_2^m, \ldots, V_{N+M}^m, \Phi_{N+M}^m]$, where N and M are the number of nodes in the tissue and in the bath, respectively (and similarly with linear systems (3.93) and (3.94)). The reasons for this choice are given in Section 4.4.2. Furthermore, the linear system (3.104) can be rewritten as

$$A\mathbf{x} = \mathbf{b} \tag{4.10}$$

with $A \in \mathbb{R}^{(N+M)\times(N+M)}$ and $\mathbf{x}, \mathbf{b} \in \mathbb{R}^{N+M}$ (and similarly with (3.93) and (3.94)). From a formal point of view, the use of an interleaved unknown ordering can be seen as the effect of applying the following permutation to the linear system (4.10):

$$QAQ^TQ\mathbf{x} = Q\mathbf{b} \tag{4.11}$$

with the orthogonal permutation matrix $Q = [q_{ij}] \in \mathbb{R}^{(N+M)\times(N+M)}$

$$q_{ij} \quad := \quad \begin{cases} 1, & i \text{ is even } \wedge\ j = \frac{i+N+M}{2} \\ 1, & i \text{ is odd } \ \wedge\ j = \frac{i+1}{2} \\ 0, & \text{otherwise} \end{cases} \quad . \tag{4.12}$$

Below we illustrate how Equation (4.11) would look for a bidomain simulation

in a hypothetical domain made of two grid points and with two processors (the solid line indicates the border between processors):

$$\left(\begin{array}{cccc} V_{11} & \phi_{11} & V_{12} & \phi_{12} \\ V_{31} & \phi_{31} & V_{32} & \phi_{32} \\ \hline V_{21} & \phi_{21} & V_{22} & \phi_{22} \\ V_{41} & \phi_{41} & V_{42} & \phi_{42} \end{array}\right) \left(\begin{array}{c} V_1 \\ \phi_1 \\ \hline V_2 \\ \phi_2 \end{array}\right) = \left(\begin{array}{c} b_1 \\ b_3 \\ \hline b_2 \\ b_4 \end{array}\right) \tag{4.13}$$

where $\{V, \phi\}_{ij}$ are the coefficients associated with unknown $\{V, \phi\}_j$ in the i-th equation.

4.4.1 Mesh load and partitioning[16]

We saw in Section 3.4 that the first stage of a simulation involves reading in from disk the mesh describing the computational domain. The standard file formats used to represent meshes in cardiac electrophysiology simulation typically consist of separate ASCII files containing a list of node coordinates (node file), a list of the nodes contained in each element (element file) and a list of the nodes contained in each surface element (face file). For large meshes these files can grow to be very large, e.g. for a mesh containing approximately 4 million nodes and 24 million elements the file sizes are 129 MB (node), 958 MB (element) and 33 MB (face). Further, variable field length in ASCII files makes random access to file impossible, so it is necessary for each process to read the three files in their entirety (sometimes more than once), determine what information it needs to retain and discard the rest.

In order to reduce the amount of data that each process is required to read, Chaste defines a binary input format. This has two consequences: the files are much smaller (and hence can be read in less time) and each entry has a fixed size (allowing random access). Therefore, it is now easy for each processor to just read a

[16]Section 4.4.1 was developed in collaboration with Dr James Southern

subset of the nodes without accessing the whole file. This is not enough to prevent a complete access to the element file, though. Given a node-based mesh partition each process needs to determine which elements it owns in order to construct the mesh. Since a process owns an element if it also owns one or more of its nodes (as described in Section 4.3.2), it is necessary for that process to do a complete pass through the element file (generally the largest of the mesh files). Thus, in order to determine element ownership a fourth (binary) mesh file was introduced for the largest meshes. This is the reverse of the element file: containing a list of which elements each node is contained in, known as the node-connectivity list (NCL) file. Each process can then access the parts of the node-connectivity file that correspond to the nodes that it owns without the need to do a complete pass. Note that for a simulation on p processes, each process can be expected to own around $1/p$th of the total number of elements — so introducing the node-connectivity file is key to making the mesh load scale.

The METIS-based mesh partitioning algorithm described in Section 4.3 requires each process to create the graph representation of the whole mesh and then call METIS. This makes the method unsuitable for partitioning meshes with an associated graph too large to fit into the memory of a single core. Furthermore, since each process calls a sequential METIS algorithm, the overall time required to obtain the partition does not decrease with number of processors (but rather increases). Modifying the algorithm to use ParMETIS, the parallel version of METIS, is relatively straightforward. However, ParMETIS partitions are designed to distribute elements evenly across processes whereas a node-based partition is desirable in our case. So, instead of using ParMETIS directly, the library is accessed via PETSc wrapper functions (in the `MatPartitioning` data type) that provide the functionality required to partition based on nodes that is lacking in the ParMETIS library itself.

4.4.2 RHS assembly

In Chaste's original design, the system RHS used to be assembled element-wise as described in Section 3.1.6:

$$\mathbf{b} = \sum_{t_i \in T} L_{t_i}^T \mathbf{b}_{t_i}\,, \tag{4.14}$$

where $\mathbf{b}_{t_i} \in \mathbb{R}^{n_{t_i}}$ is the element-wise contribution to the right-hand side of (3.104) or (3.109), which mainly involves evaluating the expression

$$\frac{\chi \mathcal{C}}{\Delta t} M \mathbf{V}_{(1)}^m + \mathbf{c}^m\,. \tag{4.15}$$

where $\mathbf{c}^m$ depends on the values of I_i and I_{ion} defined across the computational domain (see Section 3.3 for more details).

In [Pathmanathan et al., 2010], we showed that provided that I_i and I_{ion} are known point-wise at the nodes, (4.15) can be conveniently recast as

$$\frac{\chi \mathcal{C}}{\Delta t} M (\mathbf{V}_{(1)}^m + \mathbf{C}^m)\,, \tag{4.16}$$

where $\mathbf{C}^m$ is a vector with the nodal values of the source term at timestep m. This operation can therefore be implemented as a vector summation followed by a matrix-vector product yielding a speed up of factor 68 over the element-wise evaluation (4.14) (the interested reader can refer to [Pathmanathan et al., 2010] for more details about this benchmark). Another important property of this formulation is that values of $\mathbf{V}_{(1)}^m$ do not need to be explicitly communicated across processor borders since the matrix-vector product will do it implicitly.

One final consideration is that in the case of bidomain RHS assembly, expression (4.15) only accounts for the first N (out of $2(N+M)$) entries of the right-hand side of (3.104). However, its assembly requires solving $I_{\text{ion}}(\mathbf{u}^n, V^n)$ (i.e. the ionic model) at each grid point. In order to achieve good load balance in the ionic model

solution, we assign ownership of the ionic models at each grid point evenly among the available processors. With the original data layout proposed in (3.104)

$$\mathbf{b} = \begin{bmatrix} \frac{\chi \mathcal{C}}{\Delta t} M(\mathbf{V}_{(1)}^m + \mathbf{C}^m) \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{d} \end{bmatrix} \tag{4.17}$$

and the row-based partition of $\mathbf{b}$ described in Section 3.2.4

$$\mathbf{b} = \begin{bmatrix} \mathbf{b}_0 \\ \vdots \\ \mathbf{b}_{p-1} \end{bmatrix} \tag{4.18}$$

with $\mathbf{b}_i \in \mathbb{R}^{\frac{n}{p}}$ the subvector owned by processor i (assuming $n = kp$, $k \in \mathbb{N}$ for simplicity), processors owning nodes $N+1$ to $2N+2M$ would have to communicate the values of I_{ion} computed locally to the processors owning the first N rows for them to finish assembling the system right-hand side. Therefore, it makes sense to re-arrange the rows in $\mathbf{b}$ in a way that $\mathbf{C}^m$ is assembled from values of I_{ion} computed locally, therefore avoiding communication. Similarly we would like to have all the processors cooperating in the evaluation of the matrix-vector product and not only those owning the first N rows. Using an interleaved variable distribution (4.11) satisfies the previous two requirements ensuring good load balance. Finally note that this load imbalance issue does not arise in the right-hand side assembly of the monodomain linear system and therefore the previous consideration does not apply.

4.4.3 Linear system solution

The linear system arising from the FEM discretisation of the bidomain equations is sparse, symmetric, and positive semi-definite. We saw in Chapter 3 that Krylov

subspace methods are adequate for such systems. Most practitioners choose the conjugate gradient (CG) algorithm for the solution of mono/bidomain FEM linear systems [Colli-Franzone and Pavarino, 2004; Pathmanathan et al., 2010; Pennacchio and Simoncini, 2009; Plank et al., 2007]. GMRES [Whiteley, 2006] and Bi-CGSTAB [Potse et al., 2006] have also been succesfully applied. In order to study the parallel efficiency of different iterative solvers, let us consider the different computational kernels involved. In the case of Krylov subspace methods, these are: i) vector inner products, ii) `axpy` operations[17], iii) matrix-vector products, and iv) preconditioning application. `axpy` operations do not compromise parallel scalability since they can be performed without need of communication (assuming a consistent parallel distribution of all the vectors involved). Matrix-vector products require a certain degree of communication, but as we saw in Section 4.3 reduction of the matrix bandwith through the use graph-based domain decomposition techniques increases scalability. The scalability of the preconditioning step is determined by the preconditioning technique of choice (see Section 3.2.3 for a description of the following techniques): preconditioners such as point Jacobi or block Jacobi with incomplete factorisation at each sub-block do not require communication. However, whole matrix incomplete factorisation or multigrid techniques, reported to be more effective at improving the condition number of the system, are tightly coupled algorithms that require a higher degree of communication. Chapters 5 and 6 present effective bidomain preconditioners. Finally, vector inner products are communication intensive operations as they are often implemented as parallel reductions[18]. Parallel reductions are also required

[17] A vector update involving two vectors and a scalar: $\mathbf{y} = a\mathbf{x} + \mathbf{y}$, $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n, a \in \mathbb{R}$

[18] A parallel reduction is a primitive that involves a set of p parallel processes agreeing on the result of a given operation applied to data that is distributed among them. An example would be computing the minimum value in a distributed vector: each process will compute its local minimum, a reduction will be performed to find out the global minimum, followed by a broadcast operation to communicate the result of the reduction to every process. If the reduction and broadcast operations are implemented with binary trees, the cost of the parallel reduction is $O(\log(p))$. Therefore the cost of a parallel reduction increases with the number of processors. In practice, the increase in execution time can be higher than logarithmic due to network contention arising from the large volume of messages issued.

when checking for convergence if the stop criteria is based on the evolution of the l^2-norm of the residual.

Below, we will investigate how to reduce the number of parallel reductions required in order to improve scalability.

Inner product reduction

Several authors [Barrett et al., 1994; Gutknecht and Röllin, 2002] have proposed the use of the Chebyshev Iteration (CI) method (see Section 3.2.2 for a description) for the solution of symmetric linear systems avoiding inner products which often become a performance bottleneck in parallel hardware. The method requires enough knowledge about the spectrum of the preconditioned operator $P^{-1}A$ for which an interval $[a, b]$ enveloping all the eigenvalues can be defined. Formally, given the preconditioned system

$$P^{-1}A\mathbf{x} = P^{-1}\mathbf{b}\,, \tag{4.19}$$

with $P \in \mathbb{R}^{n\times n}$ and its spectral factorisation

$$P^{-1}A = Q\Lambda Q^T\,, \tag{4.20}$$

where

$$QQ^T = I, \quad \Lambda = diag(\lambda_1, \lambda_2, \ldots, \lambda_n), \quad \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n, \tag{4.21}$$

the method requires knowledge about the interval $[a, b]$ such that

$$a \leq \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n \leq b\,. \tag{4.22}$$

The following result will be useful in later discussion: let $\mathbf{r}^{(i)} = \mathbf{b} - A\mathbf{x}^{(i)}$ be the residual vectors associated with the i-th iterate generated by CI, it can be shown

that (see [Calvetti et al., 1994] or the derivation of (3.81)):

$$\mathbf{r}^{(i)} = \sum_{j=1}^{n} \alpha_j p_i(\lambda_j)\mathbf{q}_j \,, \tag{4.23}$$

where $\mathbf{q}_j \in \mathbb{R}^n, Q = [\mathbf{q}_1, \ldots, \mathbf{q}_n]$ are the eigenvectors of the system and p_i is the Chebyshev polynomial of degree i.

To the best of our knowledge, the CI method has never been succesfully applied to the solution of the bidomain equations in modern parallel hardware. In our opinion, the reasons are two-fold: i) the intrinsic difficulty of estimating $[a, b]$ for large matrices, and ii) the fact that in practice CG may converge to the solution quicker than the CI due to its superlinear convergence properties (see Section 3.2.2). This reduction in the total number of iterations required may compensate for the inferior scalability, yielding an overall shorter execution time.

In [Calvetti and Reichel, 1996] a hybrid iterative method that alternates CG and Richardson[19] iterations for the solution of symmetric positive definite linear systems is proposed. The method starts by performing m CG iterations and, thanks to the well-known relationship between CG and the Lanczos algorithm [Golub and Van Loan, 1996], it also computes a tridiagonal matrix $T_m^k \in \mathbb{R}^{m\times m}$ based on the coefficients of the CG iteration. The m eigenpairs of T_m^k, $(\hat{\lambda}_i^k, \hat{\mathbf{q}}_i^k)$, satisfy

$$\lambda_1 \leq \hat{\lambda}_1^k \leq \hat{\lambda}_2^k \leq \cdots \leq \hat{\lambda}_m^k \leq \lambda_n \tag{4.24}$$

where k is the number of times than CG has been used to compute T_m^k. After that, the interval $[a^k = \hat{\lambda}_1^k, b^k = \hat{\lambda}_m^k]$ is used to initialise the Richardson iteration and iterate until the solution is found or until slow convergence is detected. A degradation in convergence after, say, p iterations indicates that the residual $\mathbf{r}^{(m+p)}$ is orthogonal to all, or almost all, of the eigenvectors $\hat{\mathbf{q}}_i^k$, i.e. $p_{m+p}(\hat{\lambda}_i^k) = 0$ in (4.23).

[19]The Richardson iteration is a stationary iterative method for the solution of systems of linear equations [Golub and Van Loan, 1996]

At this point, m new CG iterations will be performed in order to generate a new wider interval $[a^{k+1}, b^{k+1}]$ such that

$$\lambda_1 \leq a^{k+1} \leq a^k, \qquad b^k \leq b^{k+1} \leq \lambda_n \tag{4.25}$$

The Richardson iteration is then resumed with the new interval. The hybrid scheme alternates between CG and Richardson iterations as described until a sufficiently converged solution of (4.19) is found. Note that the definition of m has been intentionally left out from the discussion, the reader can refer to [Calvetti and Reichel, 1996] for a discussion and an empirical evaluation of the optimal number.

In this Thesis, we extend this idea to the solution of linear systems with multiple — but not simultaneously available — right-hand sides, as is the case in the FEM solution of the mono/bidomain equations and other systems of PDEs including time derivatives. We propose interleaving complete CG and Chebyshev solves rather than CG and Richardson iterations within the same solve. This is motivated by the fact that efficient parallel implementations of CG and Chebyshev are readily available in third-party libraries and can be used as black boxes. In our algorithm, CG will be used for both solving the first timestep (and possibly later ones) and for computing the interval $[a^k, b^k]$ that will be used to initialise subsequent Chebyshev solves. The width of the interval — and therefore the portion of the spectrum contained — will depend on the number of CG iterations performed. Unlike [Calvetti and Reichel, 1996], we do not perform a fixed number of CG iterations but we iterate until $||\mathbf{r}^{(i)}|| < r_{tol}$, with r_{tol} being a configurable tolerance. Table 4.6 tabulates $\hat{\lambda}_1$, $\hat{\lambda}_m$ and $||\mathbf{r}^{(i)}||$ after l CG iterations for the first timestep of a bidomain simulation with the Oxford rabbit model stimulated at the apex at $t = 0$ ms.

The width of the interval $[\hat{\lambda}_1, \hat{\lambda}_m]$ is determined by the number of CG iterations performed and will dictate the convergence rate of subsequent Chebyshev solves. Underestimating it, by not including eigenpairs $(\hat{\lambda}_j, \hat{q}_j)$ associated with large values

Table 4.6 Solution residual $||\mathbf{r}^{(l)}||$ after l CG iterations, smallest eigenvalue ($\hat{\lambda}_1^l$) and largest ($\hat{\lambda}_m^l$) eigenvalue simultaneously computed with the Lanczos algorithm. First bidomain simulation timestep.

l	$\hat{\lambda}_1^l$	$\hat{\lambda}_m^l$	$\|\|\mathbf{r}^{(l)}\|\|$
1	0.696	0.696	141.10
2	0.525	0.965	39.13
3	0.481	1.048	12.49
4	0.451	1.181	4.79
5	0.425	1.336	1.62
10	0.380	1.510	0.0257
15	0.3667	2.203	0.000124
16	0.3662	2.237	0.0000529
20	0.117	2.243	0.0000168
25	0.0924	2.24485	0.00000206
30	0.0878	2.24487	0.000000273
40	0.085868	2.59049	0.00000000389

of α_j in (4.23), can lead to convergence stagnation. Overestimating it can lead to slow convergence due to the need to compute polynomials of higher degree. Table 4.7 shows the number of iterations taken by CI — initialised with $[a = \hat{\lambda}_1^l, b = \hat{\lambda}_m^l]$ for a range of values of l — for the solution of the linear system corresponding to the second timestep of the simulation described above.

Table 4.7 Number of iterations taken by CI initialised with $[a = \hat{\lambda}_1^l, b = \hat{\lambda}_m^l]$ for a range of values of l. Second bidomain simulation timestep.

l	number of iterations
3	30
5	25
16	21
19	28
22	40
54	45
58	101

It can be observed that the minimum number of iterations is achieved with $l = 16$. One may think that the optimal choice of a, b is somehow arbitrary or at best requires an expensive tuning process. However, it can be appreciated in Table

4.6 that after 16 iterations $||\mathbf{r}^{(i)}||$ goes below 10^{-4}, which corresponds to the tolerance used in the CI solver. It can therefore be concluded that the optimal choice of a, b correspond to the values of $\hat{\lambda}_1, \hat{\lambda}_m$ computed by CG when configured to solve to the same level of accuracy as later Chebyshev solves.

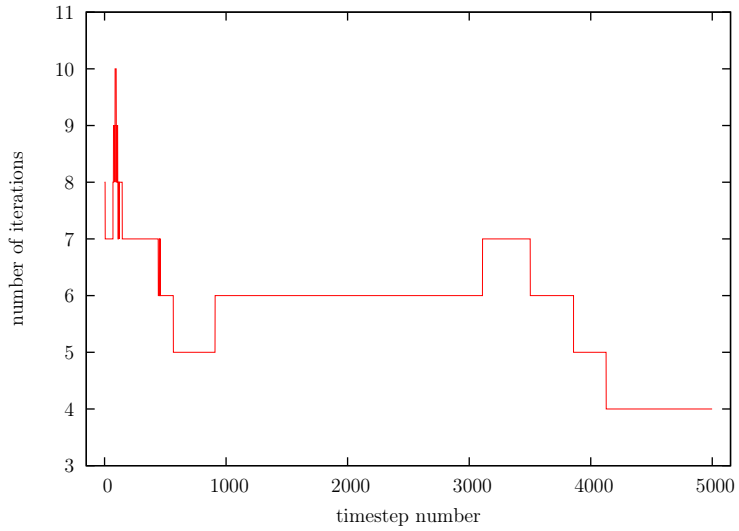
Finally, it cannot be expected that the choice of a, b will remain optimal throughout the whole simulation, especially if sudden changes to $||\mathbf{r}^{(i)}||$ occur (as a consequence of the application of stimuli, for example). Therefore, the interval $[a, b]$ needs to be re-evaluated repeatedly after a certain number of timesteps. In [Calvetti and Reichel, 1996], this is done by monitoring $||\mathbf{r}^{(i)}||$ evolution. It is our intention, however, to minimise the number of $||\cdot||$ operations since they require parallel reductions which, as we already saw, can compromise scalability at large core counts. The following subsection describes our approach.

l^2-norm reduction

Even when using an inner-product free iterative method such as CI, one is still left with a parallel reduction per iteration when the stopping criterion is based on the evolution of the l^2-norm of the residual. In this work, we propose an algorithm that overcomes this bottleneck by combining linear solves with two types of stop criteria: a) standard l^2-norm-based stop criteria and b) solves where a fixed number of iterations are performed — therefore avoiding l^2-norm computation — with a 1:s ratio (i.e. s consecutive CI solves with fixed number of iterations per each CI solve with standard stop criteria). This approach is based on the evidence that for accurate enough values of the parameters a and b, the number of iterations required by successive CI solves stays relatively constant, as shown in Figure 4.7. Note that this may require periodic re-evaluations of $[a, b]$ as already mentioned.

Sudden changes in iteration count are triggered by events such as stimuli application (both user-defined and coming from self-stimulating cells), presence in the

Figure 4.7 Number of iterations taken by CI ($r_{tol} = 10^{-6}$) at each PDE timestep for a bidomain simulation with the UCSD model.



domain of depolarisation/repolarisation wavefronts and, in general, any event inducing sudden changes in $||\mathbf{r}^{(i)}||$. For some of them, the precise moment when they occur is known at the beginning of the simulation. Therefore, it is possible to reset the 1:s ratio so that the solution is not under-approximated. For others, such as the beginning of a repolarisation wavefront, there is no easy way of anticipating them and the solution will be under-approximated until the next convergence-based solve.

A hybrid CG-Chebyshev method

Algorithm 4 summarises the technique described in the previous two subsections for the reduction of inner products and l^2-norms required by standard Krylov subspace methods. Note the two different uses of the Chebyshev Iteration linear solver: in line 7, the algorithm iterates until $||\mathbf{r}^{(i)}|| < r_{tol}$ and records the number of iterations (num_its) taken, whereas in line 9 a fixed number of iterations num_its are performed.

In order to evaluate the error introduced by the use of fixed number of iterations in Chebyshev2, we designed a benchmark consisting of 500 ms of bidomain activity following an apical stimulus on the UCSD rabbit model. Figure 4.8.(a) plots the infinity norm of the error against time for a range of values of 1:s when compared

Algorithm 4 Hybrid CG-Chebyshev method for symmetric linear systems with multiple, not simultaneously available, right-hand sides.

INPUT: System matrix $A \in \mathbb{R}^{n \times n}$, initial guess $\mathbf{x}^{(-1)} \in \mathbb{R}^n$, tolerance r_{tol}, number of timesteps n_t, 1:s ratio

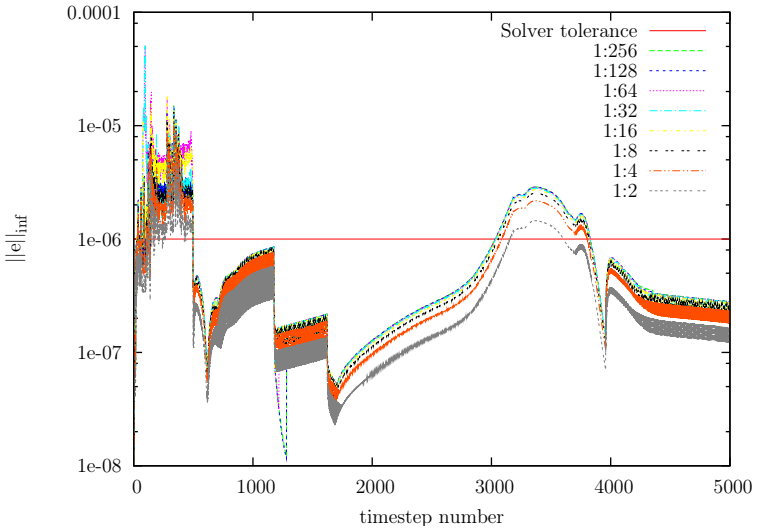
OUTPUT: $\{\mathbf{x}^{(i)}\} : A\mathbf{x}^{(i)} = \mathbf{b}^{(i)}, i = 0, 1, \ldots, n_t - 1$

1. **for** $i := 0$ **to** $n_t - 1$,
2. $\quad \mathbf{b^{(i)}}$:= assemble_rhs(i, $\mathbf{x}^{(i-1)}$)
3. $\quad$ **switch** (i mod s)
4. $\quad$ **case** 0:
5. $\quad\quad$ [$\mathbf{x}^{(i)}$, a, b] := CG(A, $\mathbf{b}^{(i)}$, $\mathbf{x}^{(i-1)}$, r_{tol})
6. $\quad$ **case** 1:
7. $\quad\quad$ [$\mathbf{x}^{(i)}$, num_its] := Chebyshev1(A, $\mathbf{b}^{(i)}$, $\mathbf{x}^{(i-1)}$, a, b, r_{tol})
8. $\quad$ **otherwise**:
9. $\quad\quad$ $\mathbf{x}^{(i)}$:= Chebyshev2(A, $\mathbf{b}^{(i)}$, $\mathbf{x}^{(i-1)}$, a, b, num_its)
10. $\quad$ **end switch**
11. **end for**

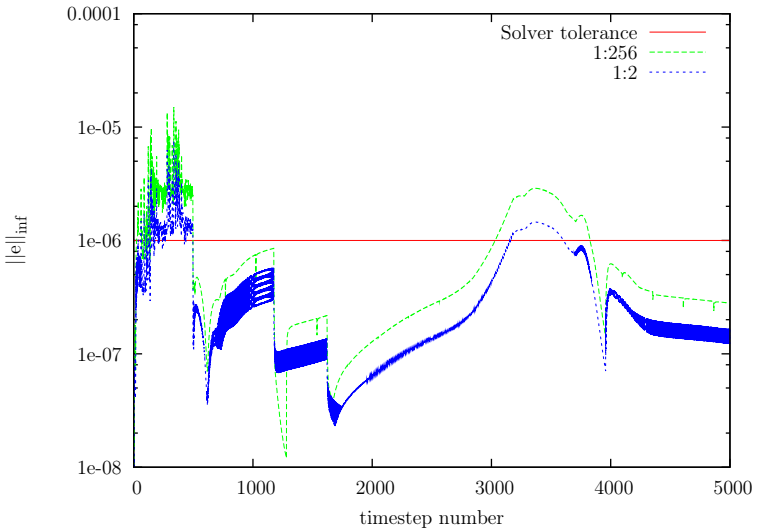
with a reference solution computed with CG. It can be appreciated that for the largest portion of the simulation, error stays within linear solve tolerance, $r_{tol} = 10^{-6}$ in this case. Only during periods when depolarisation and repolarisation wavefronts travel through the domain, is the error induced larger than r_{tol}, mainly within one order of magnitude. Once these events have ended, the error goes below r_{tol} again, which indicates that error is not accumulated over time. Figure 4.8.(b) shows how there is not a great variation in the error introduced for the different values of s studied.

Before moving on to the evaluation of the improvements presented in the current section, we will turn our attention on to some PETSc-related implementation details. We will see how PETSc can be configured to reduce the volume of communication required for the assembly of matrices and vectors by an important factor.

Figure 4.8 Error introduced by several configurations of the parameter 1:s in the hybrid CG-Chebyshev algorithm compared with a reference solution computed with CG. Panel (a) shows the error for values of s between 2 and 256. Panel (b) focuses on the configurations presenting largest and smallest error.



(a)



(b)

4.4.4 PETSc-related implementation details

As described in Section 3.2.4, Chaste uses PETSc parallel data structures wherever posible. When constructing parallel matrices and vectors, PETSc allows for data to be generated non-locally. At the last stage of construction (known as assembly), PETSc will work out the appropriate owner and migrate the data. This process requires several rounds of parallel reductions in order to ensure consistency among all the processes, even when no data is being migrated. It is possible to disable this check provided that the user makes sure that all the data is generated locally, reducing the number of parallel reductions required with an important positive impact in parallel scalability. In PETSc version 3.1 this is done for matrices with the following function call:

```
MatSetOption(lhs_matrix, MAT_IGNORE_OFF_PROC_ENTRIES, PETSC_TRUE)
```

where `lhs_matrix` is a PETSc matrix. In the case of vectors this can be done with

```
VecSetOption(rhs_vector, VEC_IGNORE_OFF_PROC_ENTRIES, PETSC_TRUE)
```

where `rhs_vector` is any PETSc vector.

In our case, we saw in Section 4.3 that the system matrix is assembled from data generated locally thanks to a small degree of mesh data replication, so that we can safely disable the check mentioned above. Similarly, when assemblying the system RHS, the portion of vector $\mathbf{C}^m$ in (4.16) owned by each process can be assembled from local data and therefore there is no need to check for non-locally generated data

4.5 Results

Having described the parallelisation of the main stages of Chaste's bidomain solver, we will now turn our attention on to the evaluation of Chaste's scalability in a large-scale HPC resource. The benchmark described in Section 4.5.1 is used to evaluate

the scalability of the code before and after the improvements presented in Section 4.4.

4.5.1 A benchmark

In order to evaluate the techniques presented on large-scale supercomputers with realistic 3D cardiac models, we designed an electrical propagation benchmark with the Oxford rabbit model (see Section 2.3 for a description) consisting of 100 ms of bidomain activity following an apical stimulus. Table 4.8 summarises the experiment details.

Table 4.8 Benchmark configuration.

simulation duration	100.0 ms
stimulus type	apical
stimulus start time	0 s
stimulus duration	0.5 ms
PDE timestep	0.01 ms
cell model	Luo Rudy 1991
ODE timestep	0.01 ms

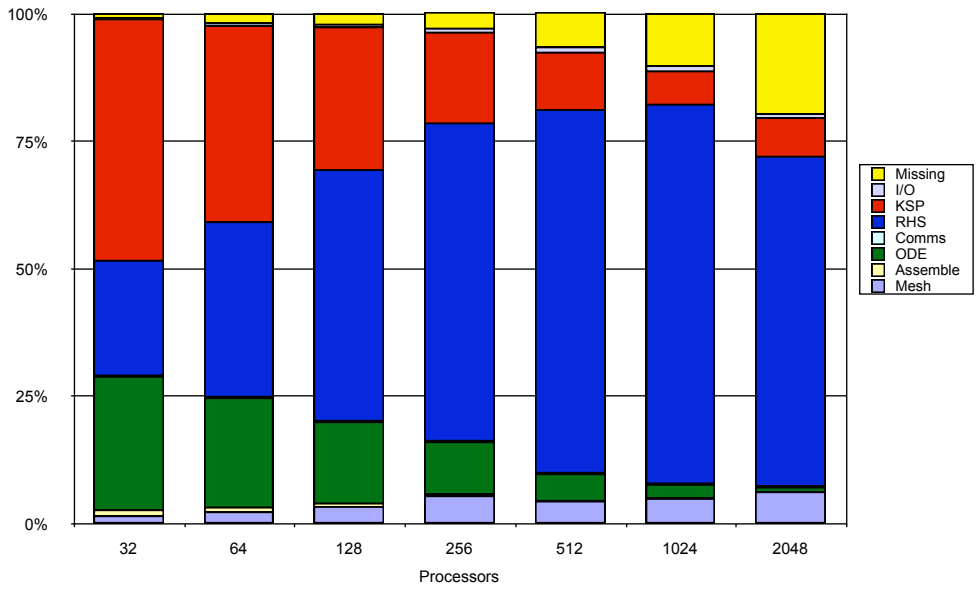
All the simulations presented in this section were run with Chaste. The following parameters were used in equations (3.86)–(3.90): $\chi = 1400$ cm^{-1}, $\mathcal{C} = 1.0$ μF/cm^2, $\sigma_i = \text{diag}(1.7, 1.7, 1.7)$ mS/cm, and $\sigma_e = \text{diag}(6.2, 6.2, 6.2)$ mS/cm, where $\text{diag}(x, y, z)$ is a 3×3 diagonal matrix with values x, y, z along the diagonal.

All the simulations presented were run in HECToR Phase 2a. This is a Cray XT4 system with 3072 compute nodes (at the time of writing). Each compute node is an AMD 2.3 GHz Opteron Barcelona quad-core. Each quad-core socket shares 8 GB of memory and a Cray SeaStar2 chip router with 6 links which are used to implement a 3D-torus network topology.

4.5.2 Original time breakdown

In this subsection, we will consider the code prior to the improvements presented in Section 4.4. Figure 4.9 presents, for an increasing number of cores, the proportion of time spent by the benchmark simulation in each of the stages described in Section 3.4.1. We will use these results as the baseline for the evaluation of the improvements presented in Section 4.4.

Figure 4.9 Original time breakdown before the improvements presented in this section. Times collected by Dr James Southern.



The first thing to note is that as core count increases, the time within the 'Missing' section starts to increase. Further profiling confirmed that it was spent outside the main stages cited earlier and therefore it was potentially redundant. More precisely, it was identified to be unnecessary synchronisation and disk access contention when writing Chaste's log files. Interestingly, this performance degradation had passed previously unnoticed when running in small size clusters and workstations

(note how it is hardly visible for $p = 32$). This is a good example of how certain operations that scale well on up to a few tens of nodes become major bottlenecks at large scale.

The next thing to note in Figure 4.9 is that the proportion of time spent in 'RHS assembly' increases with core count and starts to dominate total execution time for $p > 128$. This is a good indicator of poor scaling and therefore it was one of the first issues we addressed as described in Section 4.4.2.

It can also be seen how the portion of time spent in 'Mesh load' did not scale either. The reasons behind are two-fold: firstly, the sequential nature of the algorithm used for domain decomposition (METIS) and secondly disk access contention. These two issues were addressed as described in Section 4.4.1.

Finally, the two stages taking most of the time at $p = 32$ (i.e. 'System solution' and 'ODE solve') do not show major scaling problems or if they do so, it is not as severe as those seen in 'RHS assembly' or 'Mesh load'. This is expected from 'ODE solve', since it is an embarrassingly parallel problem and it will scale linearly provided that we generate an even distribution of the ionic models among the available processors. However, 'System solution' involves the use of tightly-coupled parallel algorithms that are likely to scale suboptimally. The same analysis presented in Figure 4.9 was repeated once the scaling issues in 'Missing', 'RHS assembly', and 'Mesh load' had been addressed, showing 'System solution' as the next target for improvement (results not shown here). The improvements are summarised in Section 4.4.3.

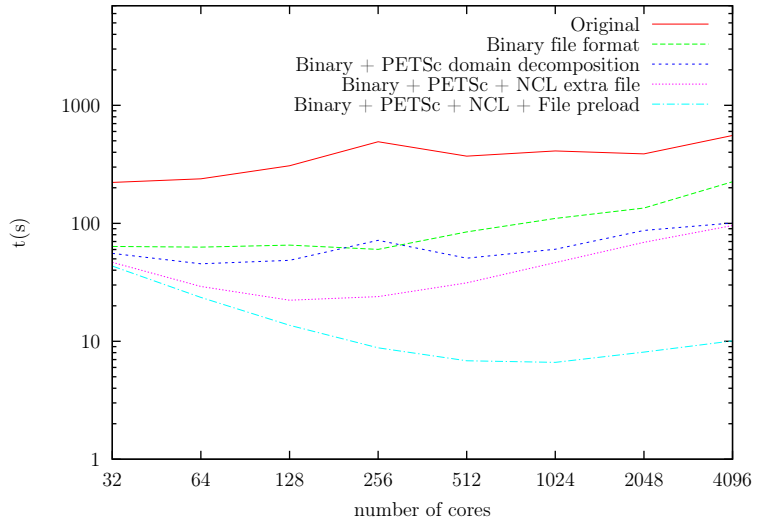
4.5.3 Scalability improvements

We will now move on to evaluating the performance improvements introduced by the optimisations described in Section 4.4.

Mesh load

Figure 4.10 presents the time taken by the mesh load stage against the number of cores for five different versions of the code. In the original version, the mesh load time increased rather than decreased with the number of processors. The reasons were two-fold: i) mesh files were entirely read by all the processors and ii) a sequential algorithm was used to compute the partitions. One would expect that in this scenario mesh load would, at best, remain constant. However, the fact that all the processors were concurrently accessing the entire set of files is likely to produce disk access contention due to the large number of read operations issued at large core counts. In the case of ii), the asymptotic cost of the algorithm is directly proportional to the number of partitions to be generated which in our case is equal to the number of processors running the simulation. In the second version presented (i.e. Binary file format), mesh read times are reduced by a factor of 4.21 on average, mainly due the fact that data representation is more compact and direct access can be performed against the node file. However, scalability still remains an issue. In the third version presented (i.e. Binary + PETSc domain decomposition), the use of PETSc-based partitioning, instead of raw calls to METIS, allows for the partitioning step to be performed in parallel, greatly reducing execution time for large core counts: a factor of 2.23 for $p = 4096$. Nevertheless, the element file is still entirely read by all the processors, introducing a sequential portion of code that still dominates total execution time. In the next version (i.e. Binary + PETSc + NCL extra file), the introduction of NCL files allows for parallelisation of the element file read step. It can be appreciated how the total mesh read time scales well up to 128 processors. However, for larger core counts mesh read time increases again due to disk access contention. In the final version (i.e. Binary + PETSc + NCL + File preload), in order to validate our hypothesis about the file access contention, a warm-up run is performed before the actual simulation being timed. The rationale

Figure 4.10 Mesh load time for different number of cores.



behind this is that the files will be cached by the distributed file system before the time the second run starts, therefore reducing latency and hiding disk access contention. When compared with the previous version, it can be appreciated how mesh load time stays fairly constant for $p = 32$ (46s vs 42s) but is greatly reduced for larger core counts: scalability is very good up to 1024 cores, achieving 92% and 80% scalability for 64 and 128 cores, respectively. We can see how total read time saturates at around 6s for $p = 1024$ and slightly increases for $p > 1024$. This is due to some portions of this stage not being parallelised and to the fact that we are solving a problem of fixed size and therefore the proportion of halo nodes and elements increases with p, making the use of large numbers of cores ineffective below a certain value of number-of-nodes to number-of-processors ratio. Hence, if we increased the mesh size in our benchmark we would probably see good scalability up to $p = 4096$ and beyond.

RHS assembly

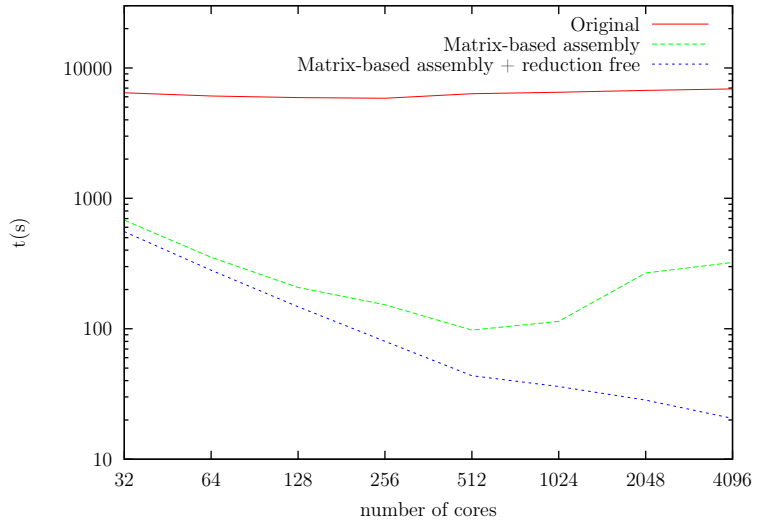
Figure 4.11 presents the time taken by the RHS assembly stage against number of cores for three different versions of the code. In the original implementation, we

can see how execution time remains constant no matter the number of cores used. We already identified this scalability problem in Figure 4.9. The reason behind it is the fact that at the beginning of the assembly stage, the distributed vector containing the solution at the previous timestep ($\mathbf{V}^m_{(1)}$ and $\mathbf{V}^m$ in the bidomain and monodomain formulation, respectively) used to be explicitly gathered at each process involved in the simulation. During earlier stages of development, this design was considered a valid alternative to programming an explicit halo value interexchange across subdomain borders for low core counts. However, this solution proved not to scale beyond a few tens of processes. Even for low core counts, the original implementation turned out to be one order of magnitude slower than later improvements. The first of these improvements (plotted as "Matrix-based assembly" in Figure 4.11) comes from the elimination of global gather operations and from the switch from element-wise assembly to matrix-based assembly described in Section 4.4.2. It can be appreciated how the solution scales well for up to 512 cores, however for larger core counts execution time goes up again. Further profiling showed that this was due to the increasing time spent performing parallel reductions during the assembly of $\mathbf{C}^m$. This scalability issue is also identified in [Tallent et al., 2009]. In order to reduce the number of global reduction operations required, the optimisation described in Section 4.4.4 was used. Note that this optimisation is only possible due to the use of the permutation described at the beginning of Section 4.4, which ensures that all the entries of $\mathbf{C}^m$ are generated locally.

System solution

At the beginning of this work, CG was Chaste's default linear solver. However, when the code was ported to HECToR other available linear solvers were evaluated. It was found that PETSc's implementation of the SYMMLQ algorithm was competitive. Finally, the hybrid CG-Chebyshev presented in Section 4.4.3 was also successfully

Figure 4.11 Right-hand side assembly time for different number of cores.



ported.

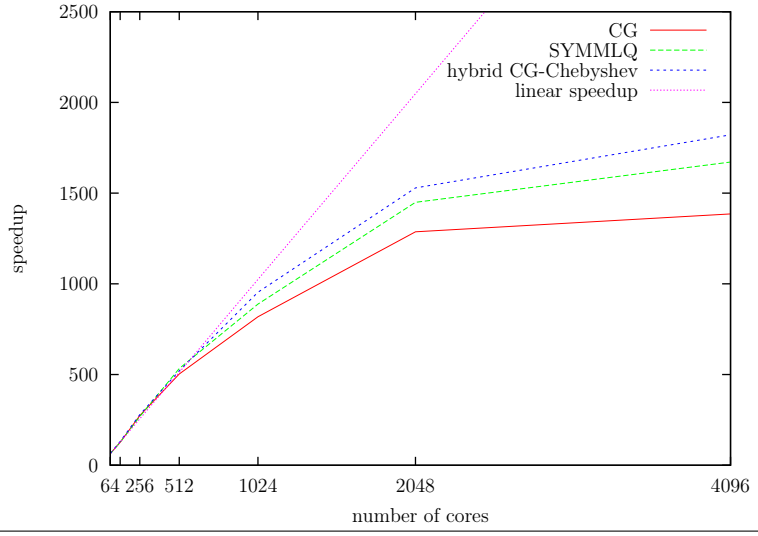
Table 4.9 presents the time taken by the system solution stage against number of cores for the three linear solvers mentioned. It can be appreciated how the SYMMLQ algorithm is faster than CG for 1024 or more cores. This is somehow unexpected since SYMMLQ needs to perform extra work to overcome the potential indefiniteness of the linear system. The magnitude of the difference in time makes us believe that this speed-up is down to implementation characteristics. Further profiling needs to be performed in order to validate this hypothesis, though. The hybrid CG-Chebyshev method is faster than any of the previous methods for all the core counts considered. For $p = 64$ the method is around 15% faster than CG, this shows that for low core counts the gains from the reduction in the number of inner products and $||\cdot||$ operations are modest and mainly due to fewer arithmetic operations being performed. For large core counts (e.g. $p = 4096$), the hybrid method is around 52% faster than CG and 35% faster than SYMMLQ. This demonstrates that as the number of cores increases the proportion of time spent performing inner products and $||\cdot||$ operations increases, mainly due to the fact that the time required to perform a global reduction is a function of p. Figure 4.12 evaluates this time

Table 4.9 Linear solve time (s) for different number of cores and the three linear solver considered

Number of processors	CG	SYMMLQ	Hybrid CG-Chebyshev 1:16
64	619	666	535
128	313	337	262
256	145	159	123
512	78.8	80	65.8
1024	48.4	48	35.9
2048	30.8	29.4	22.4
4096	28.6	25.5	18.8

reduction in terms of speedup improvement.

Figure 4.12 Parallel speedup for the three linear solvers considered and an increasing number of processors.

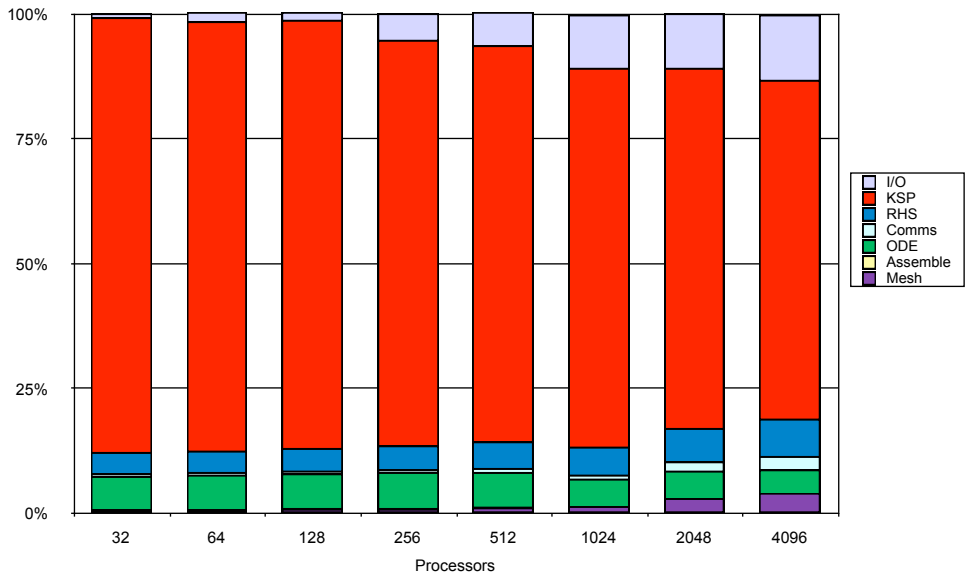


4.5.4 Final time breakdown

In this subsection, we will consider the code after the improvements presented in Section 4.4. Figure 4.13 presents, for an increasing number of cores, the proportion of time spent by the benchmark simulation in each of the stages described in Section 3.4.1. We will use these results to evaluate the overall gains derived from the improvements presented throughout this chapter.

Firstly, it can be seen how the time initially reported as 'Missing' has been suc-

Figure 4.13 Final time breakdown after the improvements presented in this section. Times collected by Dr James Southern.



cessfully removed. Secondly, the 'RHS assembly' stage has been greatly improved: it now takes no more than 8.6% of the total execution time (against 25–65% in the original time breakdown) and also scales almost linearly up to 1024 cores, degrating only slightly for 2048 and 4096 cores. Thirdly, the proportion of time spent reading in the mesh has been greatly improved, being almost unnoticeable for 32–1024 cores. However, the increasing proportion of time spent for 2048 and 4096 cores highlights the fact that the absolute time spent in this stage is actually higher than for 1024, indicating that the execution is probably suffering from file access contention. In any case, 'Mesh' time for $p = 4096$ has been reduced by a factor of 60 when compared with the initial execution. Next, we can see how 'System solution' and 'ODE' scale well and 'System solution' now takes the greatest proportion of total execution time, as one would expect from a bidomain simulation with a fairly simple ionic model such as Luo-Rudy 1991. Finally, two operations that were almost unnoticeable in the initial time breakdown increase their presence at large core counts: 'I/O' and 'Comms' (which cover writing out the simulation results and performing certain synchronisations and distributed error checks). The reasons are two-fold: i) both operations involve either synchronisation or access to resources with low degree of replication (such as Lustre[20] I/O nodes), and ii) the benchmark used consists of around 8M degrees of freedom, at $p = 4096$ the number of degrees of freedom per core is merely around 2K, which may not be enough to ensure full utilisation of all the functional units available. This situation is common when performing strong scaling analysis[21]. In fact i) is also a direct consequence of ii), operations such as 'I/O', 'Comms' or 'Mesh' and in general any task that is intrinsically sequential or with an asymptotical cost greater than $O(n/p)$ will take an increasing proportion of total time as the number of degrees of freedom per processor decreases.

[20]Lustre is the name of the parallel I/O technology used in HECToR.

[21]In the context of high performance computing there are two different notions of scalability. The first is strong scaling, which quantifies how the solution time varies with number of processors for a fixed problem size. The second is weak scaling, which quantifies how the solution time varies with number of processors for a fixed problem size per processor.

4.6 Conclusions

In this chapter we have presented effective parallelisation strategies for Chaste's bidomain solver. At the beginning of my doctorate, Chaste was not capable of running simulations with state-of-the-art geometrical models due to the lack of appropriate domain decomposition. Even for smaller models, parallel speedup used to saturate at around 10x (Figure 4.5). Section 4.3 describes the domain decomposition technique implemented in order to enable simulations with the Oxford rabbit model.

Once the code was ready to run with state-of-the-art geometrical models and the simulation requirements increased, we moved on to porting Chaste to HECToR (UK's high-end computational resource). On this machine, we were able to run simulations on two orders of magnitude more cores than before. Initial profiling highlighted a number of scalability issues only possible to identify at large scale (e.g. mesh read, RHS assembly, linear system solution). Section 4.4 describes the techniques implemented to address these issues, including novel computational and numerical techniques as well as techniques at the interface between them. Of particular interest is the novel hybrid CG-Chebyshev linear solver presented in Section 4.4.3. Although the hybrid concept was first introduced in the late 80s, we have not found any publication where the Chebyshev Iteration method has been successfully applied for the reduction of a linear system solution time in modern parallel hardware.

Figure 4.14 compares parallel speedup before and after the work presented in Section 4.4. It can be noted how in the code resulting from Section 4.3 (solid red line), speedup saturated at around 100 no matter the number of processors used. The improvements presented in Section 4.4 allowed for speedups of over 1400 and optimal or near-optimal scalability for up to 512 cores. The improvements presented in Sections 4.3 and 4.4 yield a combined speedup improvement of over 140, meaning

that Chaste's bidomain solver is now two orders of magnitude faster than at the beginning of the work described in this Thesis[22]. Table 4.10 summarises the real-time ratios achieved thanks to the improvements presented in this Chapter. The values represent an improvement respect those in Table 4.1.

Figure 4.14 Chaste's scalability for a typical bidomain simulation before (solid red line) and after (broken green line) the improvements presented in Section 4.4.

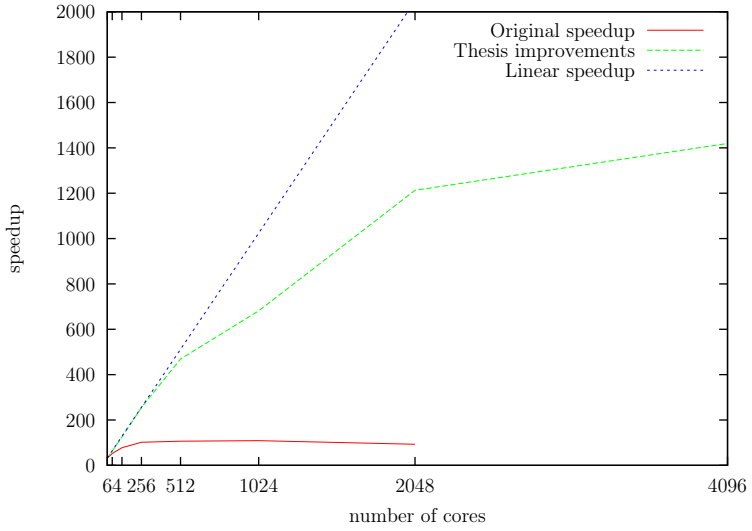


Table 4.10 Ratio between time taken to perform a simulation and the amount of time simulated.

Solver	# of cores	# of DOFS	Real-time ratio
Chaste bidomain	4096	7.4M	210
Chaste monodomain	4224	3.7M	45
Chaste monodomain	96	64K	2.8

Finally, Figure 4.13 showed how the linear system solution is now the portion of the code taking the largest proportion of the total execution time. In this chapter, we have presented ways of reducing computation and communication per iteration of linear system solution iterative methods. However, nothing has been so far mentioned about reducing the number of iterations required. We saw in Chapter 3 that the number of iterations required by iterative solvers — and therefore its total ex-

[22]This figure also includes improvements in hardware.

ecution time — is very much determined by the effectiveness of the preconditioner used. In this chapter, we have left the topic of effective parallel preconditioning for bidomain linear systems intentionally out of the discussion. In Chapters 5 and 6, we will present an in-depth study of current and novel preconditioning techniques with particular stress on parallel performance.

It can be concluded therefore that the work presented in this chapter has brought Chaste to the level of parallel performance necessary to run simulation studies with state-of-the-art whole-organ geometrical models (including human ventricular models). Chapter 7 presents the first-ever computational study of shock-induced arrhythmogenesis in the human heart. Other studies performed by the author [Bernabeu et al., 2008, 2009] and by close collaborators [Zemzemi et al., 2011] have also been possible thanks to the performance improvements presented her. Furthermore, given that Chaste is distributed as an open source software platform for bidomain simulation, the developments presented in this chapter are likely to have an immediate impact in the scientific community; making large-scale computational cardiac electrophysiology simulation more accessible, avoiding the overhead of developing multiple, often repetitive, in-house codes.

Chapter 5

Mesh-independent preconditioning for the iterative solution of bidomain linear systems

When using a semi-implicit FEM discretisation of the bidomain equations one has to deal with the solution of a large, highly sparse system of equations at each timestep. In Chapter 4, we showed that, for bidomain simulations with state-of-the-art ventricular models, the solution of this linear system is the simulation stage taking the largest proportion of time. Therefore, any performance improvement introduced in this part of the simulation is likely to have an important impact on overall performance.

In the current chapter, we develop new preconditioning techniques for iterative linear solvers in order to reduce bidomain simulation time. We identify optimal preconditioners with respect to both stimulus protocol and tissue properties. The results are supported by a comprehensive study of the mesh-dependence properties of several preconditioning techniques found in the literature. The results show that when tissue is considered isotropic and only intracellular stimuli are applied, incomplete LU factorisation remains a valid choice for current cardiac geometries.

However, when anisotropic tissue is considered or extracellular shocks are delivered — as required by our application of interest: shock-induced arrhythmogenesis — preconditioners that take into account the structure of the system minimise execution time and ensure mesh-independent convergence. The results in this chapter have been published in [Bernabeu et al., 2010a].

5.1 Introduction

The system of linear equations arising from the discretisation of the bidomain equations is highly sparse, structured, positive semi-definite, and singular. Amongst the available linear system solution algorithms (e.g. matrix-factorisation based, stationary and non-stationary iterative methods), the use of Krylov subspace (KSP) methods is common practice ([Vigmond et al., 2008], [Pennacchio and Simoncini, 2009], [Pavarino and Scacchi, 2008]). Furthermore, the use of preconditioning is known to improve performance of KSP methods. In this chapter, we will focus on the development of efficient preconditioners as a way of improving performance of bidomain simulation software. We will use CG as the linear solver, however the developments presented are also applicable to the hybrid CG-Chebyshev method presented in Chapter 4.

In the context of bidomain linear algebra, several preconditioning techniques have been proposed in recent years. They can be classified into: a) generic preconditioning techniques such as Symmetric Successive Over Relaxation [Pennacchio and Simoncini, 2002], Incomplete LU factorisation ([Whiteley, 2006]), block Jacobi [Colli-Franzone and Pavarino, 2004; Vigmond et al., 2002], Additive Schwarz [Munteanu and Pavarino, 2009; Pavarino and Scacchi, 2008; Scacchi and Pavarino, 2008], and Multigrid techniques [dos Santos et al., 2004; Pennacchio and Simoncini, 2009; Plank et al., 2007]; and b) problem-specific preconditioners that take into account the structure of the system matrix [Pennacchio and Simoncini, 2009].

[Vigmond et al., 2008] provide a comparison of several preconditioning techniques for the solution of the elliptic part of the system (after decoupling). [Pennacchio and Simoncini, 2009] compare a generic Algebraic Multigrid (AMG) technique and two problem-specific preconditioners. They also study the influence of the choice of bidomain formulation (i.e. parabolic-parabolic vs parabolic-elliptic) in the overall CG performance.

Although extensive, the literature in bidomain preconditioning presents some gaps. Firstly, preconditioners are often evaluated in 2D regular geometries [Pennacchio and Simoncini, 2009; Whiteley, 2006], 3D regular geometries [Munteanu and Pavarino, 2009; Pennacchio and Simoncini, 2002] or idealised ventricular geometries [Colli-Franzone and Pavarino, 2004; Pavarino and Scacchi, 2008]. An exception to this are the works [dos Santos et al., 2004; Plank et al., 2007; Vigmond et al., 2008] where realistic 3D geometries are considered. However, these authors only study preconditioning for the decoupled solution of the bidomain equations. [Southern et al., 2009] proved that solving the bidomain equations in a coupled manner (as described in Chapter 3) is computationally more efficient and, therefore, we prefer this method. Secondly, the mesh-dependence properties of some of the techniques proposed are not well understood. In fact, to the best of our knowledge, there is no evaluation of such property in 3D geometries for any of the techniques mentioned above.

The purpose of this chapter is two-fold. Firstly, to understand the mesh-dependence properties of some of the preconditioning techniques proposed in the literature. Secondly, to propose optimal preconditioners (with respect to mesh geometry) for the FE solution of the bidomain equations in 3D anatomically-based geometries taking into account the bidomain formulation (parabolic-parabolic vs parabolic-elliptic) and the stimulus protocol used (intracellular stimuli vs extracellular shocks).

The rest of the chapter is structured as follows. We present state-of-the-art

preconditioners for the bidomain equations in Section 5.2 and describe the benchmarks used to evaluate them in Section 5.3. Section 5.4 investigates the convergence properties of CG preconditioned with the previous techniques in the presence of intracellular stimuli and extracellular shocks, respectively. Section 5.5 analyses the mesh-dependence properties of the preconditioners considered. In Section 5.6 the effect of tissue anisotropy on preconditioner effectiveness is studied. Finally, Section 5.7 presents the conclusions of the chapter.

5.2 Bidomain preconditioning

This section presents state-of-the-art problem-specific preconditioning techniques for bidomain linear systems along with their implementation. We will consider preconditioning for both parabolic-parabolic and parabolic-elliptic bidomain linear systems (see Section 3.3 for a description).

5.2.1 Block preconditioners for bidomain linear systems

Within this section, for notational ease, we will only consider the system given by (3.93) (i.e. parabolic-elliptic bidomain without bath). This is a linear system of the form $A\mathbf{x} = \mathbf{b}$, with $\mathbf{x}, \mathbf{b} \in \mathbb{R}^{2N}$, $A \in \mathbb{R}^{2N \times 2N}$, where

$$A = \begin{bmatrix} A_1 & B^T \\ B & A_2 \end{bmatrix} \tag{5.1}$$

and $A_1 := \frac{\chi \mathcal{C}}{\Delta t} M + K[\sigma_i]$, $A_2 := K[\sigma_i] + K[\sigma_e]$, and $B := K[\sigma_i]$.

We may factor A into the form

$$A = \begin{bmatrix} I & 0 \\ BA_1^{-1} & I \end{bmatrix} \begin{bmatrix} A_1 & 0 \\ 0 & S \end{bmatrix} \begin{bmatrix} I & A_1^{-1}B^T \\ 0 & I \end{bmatrix} =: LDU$$

where $S := A_2 - BA_1^{-1}B^T$, is the Schur complement of A_1. Hence, in factored form, we have

$$A^{-1} = \begin{bmatrix} I & -A_1^{-1}B^T \\ 0 & I \end{bmatrix} \begin{bmatrix} A_1^{-1} & 0 \\ 0 & S^{-1} \end{bmatrix} \begin{bmatrix} I & 0 \\ -BA_1^{-1} & I \end{bmatrix}.$$

The main difficulty in executing this inverse, or approximating it, lies in finding a cheap reliable approximation to S^{-1}. In [Pennacchio and Simoncini, 2009], the approximation $S \approx A_2$ was considered, leading to the preconditioner

$$\mathcal{P}_{LDU}^{-1} := \begin{bmatrix} I & -A_1^{-1}B^T \\ 0 & I \end{bmatrix} \begin{bmatrix} A_1^{-1} & 0 \\ 0 & A_2^{-1} \end{bmatrix} \begin{bmatrix} I & 0 \\ -BA_1^{-1} & I \end{bmatrix} \tag{5.2}$$

where A_2^{-1} represents the pseudoinverse of the singular matrix A_2. This inverse is not computed explicitly but approximated and acts upon a consistent right-hand side (see Section 3.3.4 for more details). Along with this choice, the computationally cheaper approximation

$$\mathcal{P}_{BD}^{-1} := \begin{bmatrix} A_1^{-1} & 0 \\ 0 & A_2^{-1} \end{bmatrix} \tag{5.3}$$

was also considered.

This leads to the preconditioned matrix:

$$\mathcal{P}_{LDU}^{-1}A = \begin{bmatrix} I & A_1^{-1}B^T(I - A_2^{-1}S) \\ 0 & A_2^{-1}S \end{bmatrix}.$$

We see that the eigenvalues of the preconditioned system $\mathcal{P}_{LDU}^{-1}A$ are $\Lambda(\mathcal{P}_{LDU}^{-1}A) = \{1\} \cup \Lambda(A_2^{-1}S)$, with at least N equal to one. Thus, we are left to calculate $\Lambda(A_2^{-1}S)$.

$$\begin{aligned} \Lambda(A_2^{-1}S) &= \Lambda(I - A_2^{-1}BA_1^{-1}B^T) \\ &= \{1 - \Lambda(A_2^{-1}BA_1^{-1}B^T)\} \end{aligned}$$

For $\mathcal{P}_{LDU}^{-1}$ to be mesh-independent we need $\Lambda(A_2^{-1}BA_1^{-1}B^T) \in (0, \beta)$ with $\beta < 1$ and β not being a function of the mesh characteristics. For the lower bound we see that the eigenvalues of $A_2^{-1}BA_1^{-1}B^T$ satisfy

$$\begin{aligned} BA_1^{-1}B^T\mathbf{x} &= \lambda A_2\mathbf{x} \qquad\qquad \mathbf{x} \neq \mathbf{c} \\ \lambda &= \frac{\mathbf{x}^T BA_1^{-1}B^T\mathbf{x}}{\mathbf{x}^T A_2\mathbf{x}}. \end{aligned}$$

Since A_1 and A_2 are positive semi-definite, $\lambda \geq 0$. For the upper bound (similarly to Lemma 4.3 [Pennacchio and Simoncini, 2009]) we have[1]

$$\begin{aligned} \frac{\mathbf{x}^T BA_1^{-1}B^T\mathbf{x}}{\mathbf{x}^T A_2\mathbf{x}} &= \frac{\mathbf{x}^T K[\sigma_i](\frac{\chi\mathcal{C}}{\Delta t}M + K[\sigma_i])^{-1}K[\sigma_i]^T\mathbf{x}}{\mathbf{x}^T(K[\sigma_i] + K[\sigma_e])\mathbf{x}} \\ &\leq \frac{\mathbf{x}^T K[\sigma_i]\mathbf{x}}{\mathbf{x}^T(K[\sigma_i] + K[\sigma_e])\mathbf{x}} < 1 \end{aligned}$$

since $K[\sigma_i] < K[\sigma_i] + K[\sigma_e]$.

Hence the spectrum of $\mathcal{P}_{LDU}^{-1}A$ is bounded independently of the mesh characteristics and CG will converge at a rate independent of the mesh size. Equivalent bounds can be derived for linear systems (3.94) and (3.104) (i.e. parabolic-parabolic bidomain without bath and parabolic-elliptic bidomain with bath).

5.2.2 Inexact Preconditioning

The proposed preconditioners rely upon providing a reliable and practical method for calculating the action of both A_1^{-1} and A_2^{-1}. Recall, in (5.1) A_1 arises from a backward Euler finite element approximation of the parabolic differential operator $\chi\mathcal{C}\frac{\partial \cdot}{\partial t} - \nabla \cdot (\sigma\nabla\cdot)$, and A_2 arises from finite element approximation of the elliptic differential operator $-\nabla \cdot (\sigma\nabla\cdot)$. Hence, in terms of the dependence on discrete variables, we have $A_1 \equiv A_1(h, \Delta t)$ and $A_2 \equiv A_2(h)$.

We consider two types of approximation to the action A_1^{-1} and A_2^{-1}: i) the use

[1] We use the notation $A \leq B$ if $\forall \mathbf{v}, \mathbf{v}^T A\mathbf{v} \leq \mathbf{v}^T B\mathbf{v}$

of an incomplete LU factorisation with no fill-in (ILU(0)), and ii) one V-cycle of Algebraic Multigrid (AMG) (see Section 3.2.3 for more details). Note, in terms of construction and cost per iteration ILU(0) is computationally cheaper than AMG.

When applying ILU(0) as a preconditioner to the matrix $A_1(h, \Delta t)$ it is known that the eigenvalues are bounded in a box dependent on the ratio $\sigma_i \Delta t/h^2$ [Wathen, 1988]. In summary, when this ratio is small the preconditioner is reasonable and when it is large the approximation is poor. Hence, with respect to reducing mesh size h and timestep Δt at the same rate, typical for the implicit backward Euler time discretisation, we see that this preconditioner does not provide a reliable scheme. However, in the case of realistic cardiac geometries, the minimum value of h is limited by the current state-of-the-art in imaging and meshing technologies. In Section 5.4.1 we will show that for current ventricular models, the ratio $\sigma_i \Delta t/h^2$ equations remains reasonably low. Furthermore, we will provide a lower bound for h for which ILU(0) outperforms mesh-independent techniques.

The application of AMG for both the A_1 and A_2 matrices is known to provide an excellent preconditioner, see [Silvester et al., 2001; Wesseling, 1992]. The only downside is that the speed of constructing such a preconditioner is considerably slower than ILU (0), although it does scale linearly with refinement.

5.2.3 Implementation

All the simulations presented in this chapter were run with Chaste. The parabolic-elliptic formulation of the bidomain equations has been the default in Chaste since the first bidomain solver was written. For this work a parabolic-parabolic solver was implemented. Extra tests were written to check that solutions are consistent no matter the solver used. The following parameters were used in equations (3.84)–(3.90) and (3.95)–(3.99): $\chi = 1400$ cm^{-1}, $\mathcal{C} = 1.0$ μF/cm^2, $\sigma_i = 1.75\mathbf{I}$ mS/cm, and $\sigma_e = 7.0\mathbf{I}$ mS/cm, where $\mathbf{I} \in \mathbb{R}^{3\times 3}$ is the identity matrix. The resting length constant

of the tissue is defined as $\lambda = \sqrt{\frac{\sigma R_m}{\chi}}$ [Roth and Krassowska, 1998], where R_m is the membrane resistance and $\sigma = \frac{\sigma_i \sigma_e}{\sigma_i + \sigma_e}$ the tissue conductivity. With the parameter values listed above and a membrane resistance of 2.7 Ωcm^2[Sampson and Henriquez, 2005], $\lambda = 519$ μm which is larger than all the spacestep values considered in our study.

Chaste uses the implementation of CG and ILU(0) found in PETSc. AMG preconditioning of the whole system is performed with one V-cycle of the BoomerAMG algorithm (see Section 3.2.3 for more details). Preconditioners (5.2) and (5.3) were implemented with PETSc's PCSHELL interface, with inverses of matrix sub-blocks approximated with one V-cycle of BoomerAMG. Source code is available as part of Chaste's standard release. Table 5.1 summarises the 8 possible combinations of formulation and preconditioner.

Table 5.1 Combinations of bidomain formulation and preconditioner studied.

Code	Bidomain formulation	Preconditioner
PE-ILU	parabolic-elliptic	ILU(0)
PE-AMG	parabolic-elliptic	AMG
PE-BD	parabolic-elliptic	BD
PE-LDU	parabolic-elliptic	LDU
PP-ILU	parabolic-parabolic	ILU(0)
PP-AMG	parabolic-parabolic	AMG
PP-BD	parabolic-parabolic	BD
PP-LDU	parabolic-parabolic	LDU

Left preconditioning is used throughout this work. Therefore, the system to solve can be rewritten as $P^{-1}A\mathbf{x} = P^{-1}\mathbf{b}$, where P^{-1} is the preconditioner. The l^2-norm of the residual at the end of the i-th CG iteration becomes

$$||\mathbf{r}_p^i||_2 = ||P^{-1}\mathbf{b} - P^{-1}A\mathbf{x}^i||_2 = ||P^{-1}\mathbf{r}^i||_2.$$

Different choices of P^{-1} will, therefore, affect the value of $||\mathbf{r}_p^i||_2$. Since we want to fairly compare multiple preconditioners, we configure PETSc to use the unprecondi-

tioned residual, $||\mathbf{r}^i||_2$, for convergence testing. This introduces an overhead at the end of each iteration since $||\mathbf{r}^i||_2$ has to be explicitly computed. Unless otherwise stated, a relative tolerance of 10^{-10} and PETSc's default convergence criteria is used in all the experiments.

5.3 Benchmarks

In the previous section we presented two problem-specific preconditioning techniques for bidomain linear systems, namely BD and LDU, and proved the mesh-independent properties of LDU. In the current section, we will describe the benchmarks used to evaluate them along with other generic preconditioners described in Section 3.2.3. We are interested in two aspects: their performance in experiments with realistic ventricular geometries and their mesh-dependence properties.

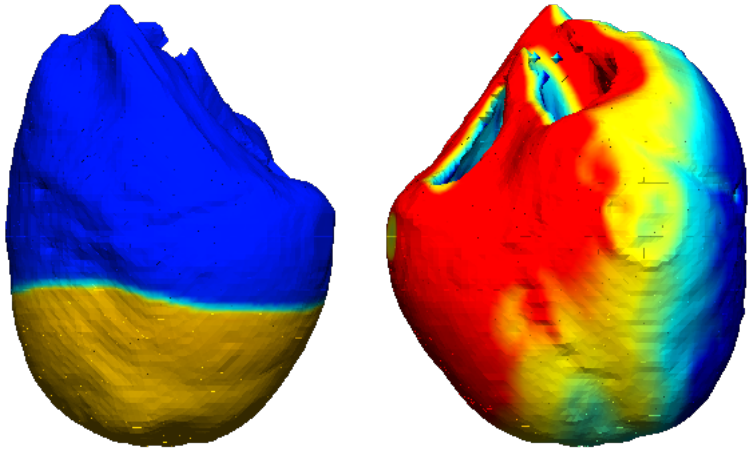
5.3.1 Practical preconditioners for realistic 3D geometries

In this benchmark, we will use the high, low and very low resolution version of the Oxford rabbit mesh both with and without bath (see Section 2.3 for a description). We will consider two stimulation protocols: a) intracellular stimulus in the apex region (delivered at a small region of 4 mm $\times$ 4 mm $\times$ 0.2 mm), and b) monophasic squared extracellular shock through electrode plates located at the boundaries of the bath parallel to the saggittal plane. Figure 5.1 shows typical cardiac activity triggered by both protocols. Table 5.2 summarises the configuration of this experiment.

5.3.2 Mesh-dependence study

For the mesh-dependence evaluation of the preconditioners, a set of 3D tissue slabs were generated. The original volume is a 0.14 $\times$ 0.14 $\times$ 0.14 cm cube, meshed as a

Figure 5.1 (a) Wavefront travelling along cardiac geometry after apical stimulus. (b) Polarised distribution of potential after extracellular shock.



(a) (b)

Table 5.2 3D experiment configuration.

	(a) Intracellular stimulus	(b) Extracellular shock
simulation duration	2.0 ms	2.0 ms
stimulus type	apical	squared shock
stimulus start time	0.2 ms	0.0 ms
stimulus duration	0.5 ms	0.5 ms
PDE timestep	0.01 ms	0.01 ms
cell model	Luo Rudy 1991	Luo Rudy 1991
ODE timestep	0.01 ms	0.01 ms

regular grid with $h \in \{\frac{0.035}{i+1}\}$ cm, $i = 0, \ldots, 4$. The number of nodes and elements in each mesh is $(2^{i+2} + 1)^3$ and $2^{3(i+2)}$ respectively. An intracellular stimulus is applied to one of the faces of the cube and the convergence history is analysed in the timesteps following the stimulation. Table 5.3 summarises the configuration of the mesh-dependence experiment.

Table 5.3 Mesh-dependence experiment configuration.

simulation duration	0.1 ms
stimulus type	all cells on the $x = 0$ face
stimulus start time	0 ms
stimulus duration	0.1 ms
PDE timestep	0.01 ms
cell model	Luo Rudy 1991
ODE timestep	0.01 ms

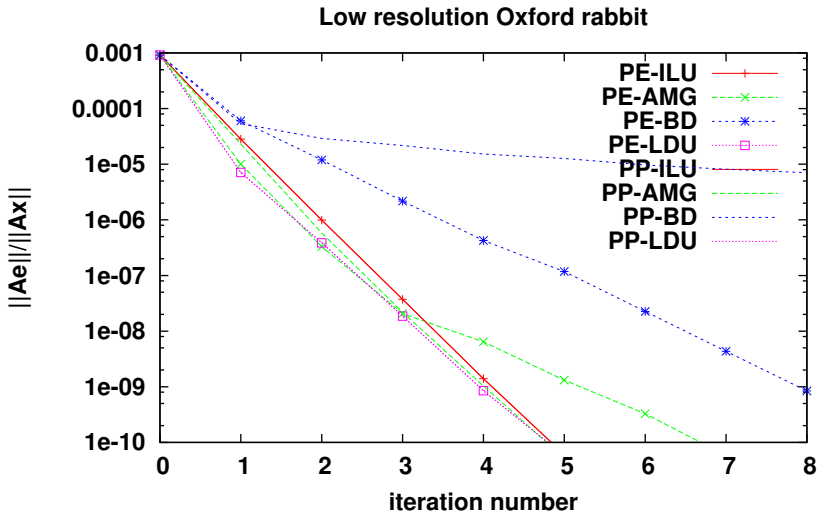
5.4 Convergence in 3D realistic geometries

5.4.1 Intracellular stimulus

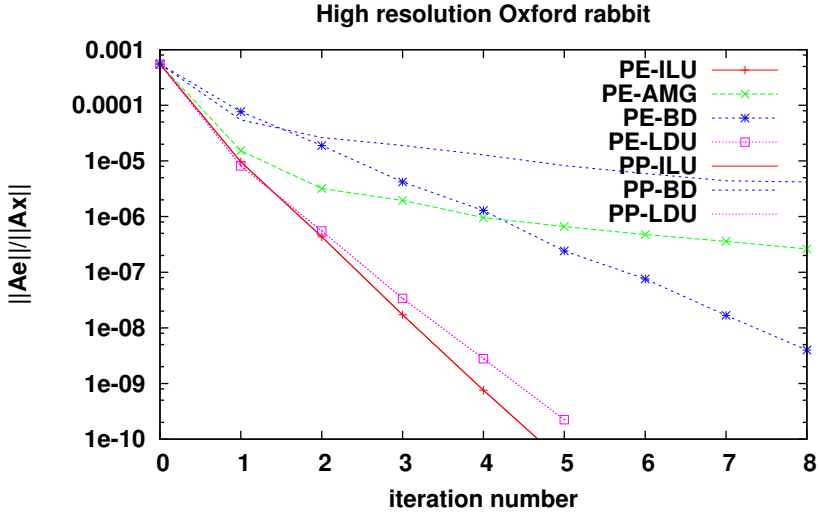
In this subsection, we evaluate the convergence properties and performance of the eight solvers presented in Table 5.1 with the intracellular stimulus benchmark described in Section 5.3.1. Since the linear systems arising have different size, the relative residual norm ($||\mathbf{r}^i||_2/||\mathbf{b}||_2$) is used for like-for-like comparison. Figure 5.2 plots the convergence history for the timestep starting at $t = 0.2$ ms, i.e. when the intracellular stimulus is applied.

Table 5.4 compares the computational efficiency of all the methods considered when solving 2.0 ms of bidomain activity in the low and high resolution mesh, respectively. The second column shows the average execution time taken by the 200 linear system solutions corresponding to the 200 timesteps simulated (setup time is not considered). The stimulus was applied after 0.2 ms in order to study how the

Figure 5.2 Convergence history at onset of intracellular stimulus ($t = 0.2$ms) for different preconditioners in the two realistic geometries studied. In both panels PE-ILU and PP-ILU as well as PE-LDU and PP-LDU overlap.



(a)



(b)

absence of cardiac activity affects convergence. The third column reports minimum, maximum, and average number of iterations throughout the 2.0 ms of simulation for each preconditioner.

Table 5.4 Performance comparison - low (top panel) and high (bottom panel) resolution mesh without bath.

solver	average solution time	min/ave/max number of iterations
PE-ILU	3.75 s	3/4.81/5
PE-AMG	33.04 s	3/7.01/8
PE-BD	26.83 s	4/9.375/10
PE-LDU	27.71 s	2/4.76/6
PP-ILU	3.92 s	3/4.81/5
PP-AMG	35.47 s	3/4.81/5
PP-BD	quit after 100 it	quit after 100 it
PP-LDU	27.71 s	2/4.76/5

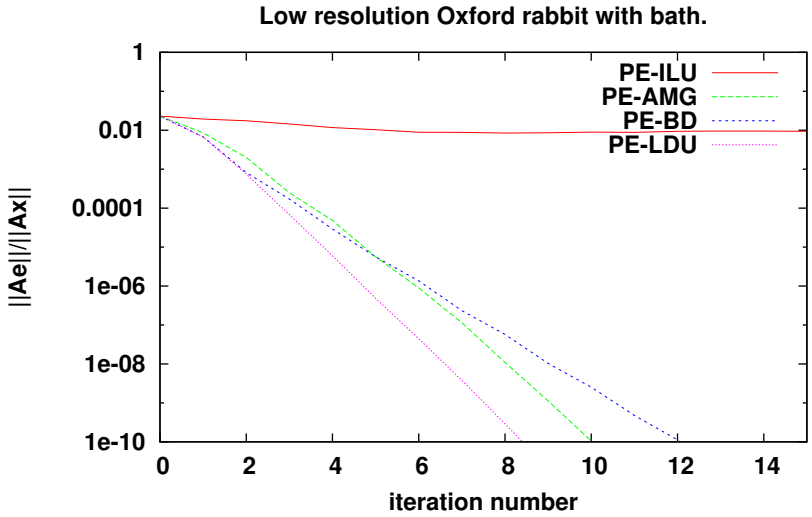
solver	average solution time	min/ave/max number of iterations
PE-ILU	25.2 s	2/3.8/5
PE-AMG	164.34 s	2/51/84
PE-BD	118.2 s	4/7/10
PE-LDU	114.8 s	2/3.8/5
PP-ILU	25.4 s	2/3.8/5
PP-AMG	crashes	crashes
PP-BD	quit after 100 it	quit after 100 it
PP-LDU	117.6 s	2/3.8/5

5.4.2 Extracellular shock

In this subsection, we study how the application of an extracellular shock affects the convergence properties and performance of CG. The extracellular stimulus benchmark described in Section 5.3.1 is used. Figure 5.3 shows convergence histories for four combinations of bidomain parabolic-elliptic formulation and preconditioner in the low resolution mesh. Results correspond to the timestep starting at $t = 0$ ms, i.e. when the extracellular shock is applied.

The computational efficiency of the four methods on this problem is compared in Table 5.5. We clearly see that ILU(0) is no longer a suitable preconditioner once

Figure 5.3 Convergence history at onset of extracellular shock ($t = 0.0$ms) for different preconditioners in the low resolution cardiac mesh with bath.



an extracellular shock is applied. This is discussed further in Section 5.7.

5.5 Mesh-dependence study

In this section, we study the mesh-dependence properties of the different solvers considered with the benchmark described in Section 5.3.2.

5.5.1 ILU(0)

Although the range of spatial stepsizes considered in Figure 5.2 are not wide enough to display it, it is known that ILU(0) is mesh-dependent (see [Wathen, 1988]). The results of the mesh-dependence study in Figure 5.4 also confirm it. However, as we already saw in Figure 5.2, the iteration count stays below 6 iterations for the values of h typical in current state-of-the-art cardiac models (i.e. a few hundred μm as shown in Table 2.1). Publications [Colli-Franzone and Pavarino, 2004; Pennacchio and Simoncini, 2009; Scacchi and Pavarino, 2008] can be found where this practical restriction is not taken into account and average edge lengths well below this limit are

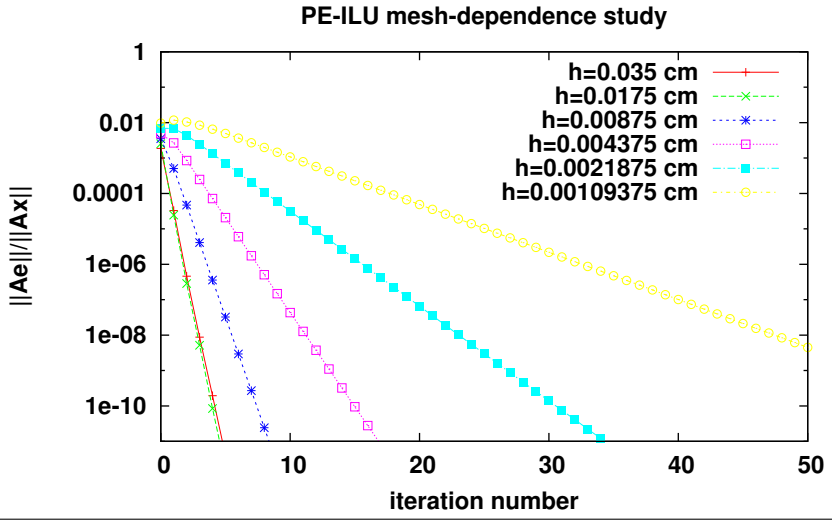
Table 5.5 Performance comparison - Very low (top panel), low (middle panel), and high resolution (bottom panel) mesh with bath.

solver	ave solution time	min/ave/max # of its
PE-ILU	quit after 100 it	quit after 100 it
PE-AMG	6.61 s	3/5.73/9
PE-BD	9.14 s	4/8.54/12
PE-LDU	18.85 s	5/4.86/18

solver	ave solution time	min/ave/max # of its
PE-ILU	quit after 100 it	quit after 100 it
PE-AMG	66.4 s	3/6.66/11
PE-BD	52.58 s	4/9.45/13
PE-LDU	53.55 s	2/5.65/9

solver	ave solution time	min/ave/max # of its
PE-ILU	quit after 100 it	quit after 100 it
PE-AMG	crashes	crashes
PE-BD	655.6 s	16/17/19
PE-LDU	660 s	10/11/13

Figure 5.4 Mesh-dependence study for PE-ILU. Convergence at onset of stimulus ($t = 0.0$ ms). PP-ILU displays the same behaviour.



considered. To the best of our understanding, this is one of the reasons why ILU(0) preconditioning is considered impractical for bidomain simulation with intracellular stimulus. Therefore, we believe that ILU(0) will remain a practical choice unless the resolution of current cardiac models increases by two orders of magnitude.

5.5.2 AMG

Several AMG algorithms have been successfully applied to the solution of the bidomain equations (BoomerAMG and Pebbles [Vigmond et al., 2008], and PIFISS [Pennacchio and Simoncini, 2009]). However, little is known about how implementation details affect performance when applied to the bidomain equations. Considerations such as how coarser grids are constructed and solutions are projected between levels have an impact in the overall performance of the method.

In our case, we have identified very different behaviour of the same preconditioner applied to two spatial discretisations of the same domain: Figure 5.2(a) shows PE-AMG converging to machine precision in 15 iterations with the low resolution mesh, whereas Figure 5.2(b) shows PE-AMG only being able to reduce the relative residual norm by 3 orders of magnitude after 40 iterations and PP-AMG failing due to memory requirements going beyond main memory size, with the high resolution mesh.

In order to investigate this situation further, we ran the mesh-dependence study described in Section 5.3 with PE-AMG and PP-AMG. The results show that in both cases iteration count is inversely proportional to h (see, for instance, Table 5.4), which confirms the mesh-dependent nature of the solvers.

We do not see PE-AMG failing to converge on any of the regular grids studied (no matter the level of refinement) as we saw in Figure 5.2(b). This suggests that its performance is related to particular characteristics of the mesh and not only to h.

5.5.3 Block preconditioners

The convergence properties of BD and LDU are analysed in [Pennacchio and Simoncini, 2009]. The authors conclude that the convergence rate of both methods is mesh-independent if good enough approximations of the inverses involved are provided. Figures 5.5 and 5.6 show mesh-dependence study for BD and LDU.

Figure 5.5 Mesh-dependence study for PE-BD. Convergence at onset of stimulus ($t = 0.0$ ms). PP-BD displays the same trends although larger numbers of iterations.

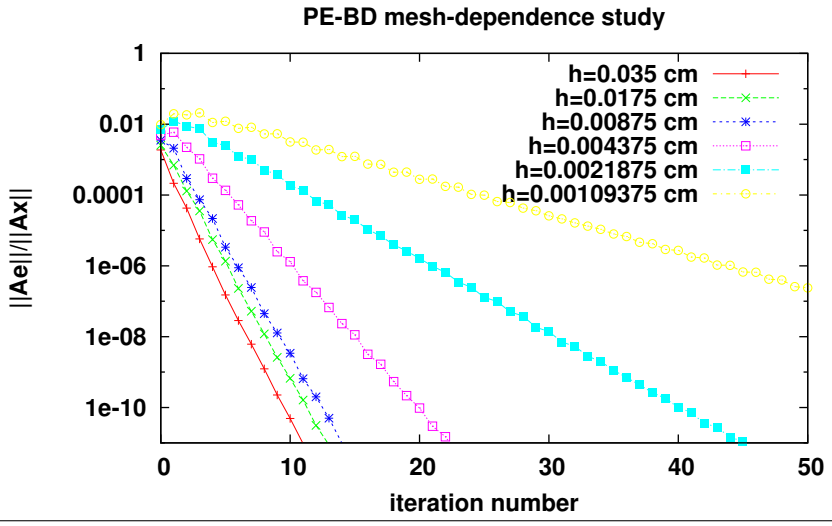
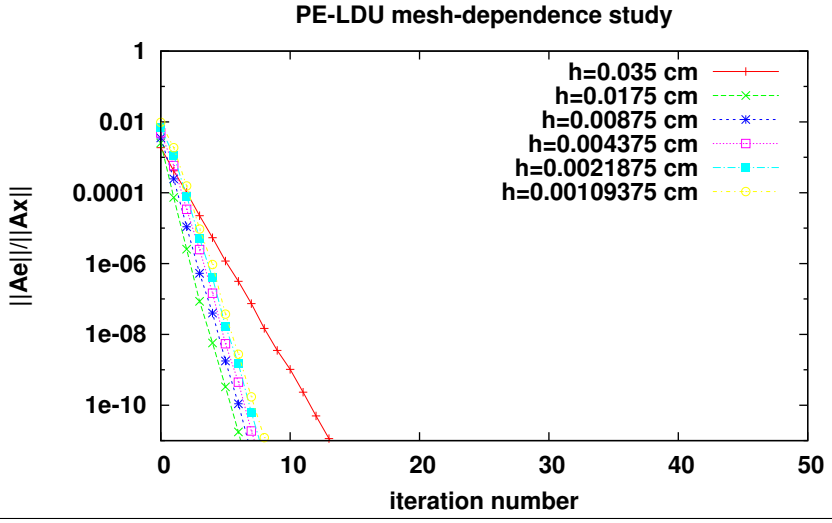


Figure 5.6 Mesh-dependence study for PE-LDU. Convergence at onset of stimulus ($t = 0.0$ ms). PP-LDU displays the same trends although larger numbers of iterations.



LDU shows mesh-independent convergence whereas BD does not. BD has been reported to be mesh-independent in [Pennacchio and Simoncini, 2009], this inconsistency requires further investigation and might be considered as future work. We observe that both preconditioners perform better when applied to the parabolic-elliptic formulation of the bidomain equations.

5.6 The anisotropic case

The benchmarks described in Section 5.3 assumed tissue isotropy for simplicity. This assumption was taken in order to avoid the impact that the ratio between conductivities in different directions has on the spectrum of the preconditioned systems (3.94) and (3.103), and only consider how h affects it (problem independence vs. mesh independence). However, cardiac tissue is known to be anisotropic in nature. In fact, recent experimental studies [Hooks et al., 2007] showed that cardiac tissue is orthotropic (i.e. has different conductivity values along the fibre, perpendicular to the fibre in the plane of the muscle layer, and normal to the muscle layer). Nevertheless, the choice of the appropriate set of coefficients is not trivial due to inconsistencies in the literature. An exhaustive comparison is outside the scope of this Thesis (see [Sebastian et al., 2008] for a survey). We will, therefore, only compare some of the values reported in [Sebastian et al., 2008] and the more recent measurements by [Hooks et al., 2007]. The first work only distinguishes between conductivity along the fibre and transversal to it, with ratios ranging from 10.76:1 to 5.6:1 for intracellular conductivity and from 2.58:1 to 1.5:1 for extracellular conductivity. The second describes tissue as fully orthotropic with ratio 4:2:1 for the bulk conductivity.

Table 5.6 summarises iteration count and execution time for the benchmark described in Table 5.3 with the geometry described in Section 5.3.2 meshed at $h = 0.004375$cm and different conductivity coefficients ($h = 0.0021875$cm shows the same behaviour). The PE-ILU and PE-LDU solvers are compared. PE-ILU

Table 5.6 Effect of anisotropy ratio choice in PE-ILU (top panel) and PE-LDU (bottom panel) performance. Number of iterations at onset of stimulus ($t = 0.0$ ms).

σ_e \ σ_i	1:1:1	4:2:1	8:4:1	16:8:1
1:1:1	18 (0.81 s)	78	79	79 (3.18 s)
4:2:1	76	13	75	78
8:4:1	82	82	11	79
16:8:1	81 (3.24 s)	83	83	8 (0.42 s)

σ_e \ σ_i	1:1:1	4:2:1	8:4:1	16:8:1
1:1:1	8 (1.67s)	8	8	8 (1.73s)
4:2:1	11	8	8	8
8:4:1	13	9	8	9
16:8:1	17 (3.56s)	11	10	11 (2.42s)

is more susceptible to anisotropic coefficients than PE-LDU. However, when the same ratios are considered for both intracellular and extracellular conductivity the increase in anisotropy ratio has a positive impact on PE-ILU efficiency (see decrease in iteration number along the diagonal of Table 5.6 top panel). Finally, our experimental results suggest that when very different anisotropy ratios are applied to the two sets of conductivities PE-LDU outperforms PE-ILU. Therefore, the choice of preconditioner in this scenario becomes empirical although the user can follow the previous guidelines as an initial guess.

5.7 Discussion and conclusions

In this chapter, we studied the convergence properties of multiple solvers applied to the solution of linear systems arising from bidomain simulations with realistic geometries. A combination of four preconditioners and two formulations of the bidomain equations were considered. Our results suggest that when only an intracellular stimulus is used, the low iteration count of CG preconditioned with ILU(0) added to the relative low computational cost of the technique makes ILU(0) the

most efficient preconditioner in our study. The choice of bidomain formulation (i.e. parabolic-parabolic vs parabolic-elliptic) does not affect its convergence rate. Our convergence analysis illustrated the problem-dependent nature of ILU(0). However, it also highlighted that if h is kept within the range of values corresponding to current state-of-the-art cardiac models (125–500 μm) the iteration count stays in the order of a few tens of iterations.

Among all the combinations tested, only PE-LDU and PP-LDU display mesh-independent convergence. The iteration count of PE-LDU is lower than its PP counterpart. BD also performs better when applied to PE. However, full AMG shows the opposite behaviour. These results are consistent with [Pennacchio and Simoncini, 2009]. Despite the mesh-independence of PE-LDU, its higher cost per iteration makes it more efficient than PE-ILU only for $h \lesssim 1$ μm. This value of h is two orders of magnitude smaller than in current realistic geometries and therefore not currently relevant. When considering anisotropic tissue, this result holds when the same anisotropy ratios are used for both intracellular and extracellular conductivity. However, when different ratios are considered in each of the spaces PE-LDU significally outperforms PE-ILU.

In the presence of extracellular shocks, the iteration count of CG preconditioned with ILU(0) increases dramatically. In our experiments, convergence was not reached within the first hundred iterations and therefore simulations were aborted. This observation is consistent with [Colli-Franzone and Pavarino, 2004] and [Vigmond et al., 2008]. However, in those publications results are generalised to any stimulus protocol. In this context the use of more sophisticated preconditioning techniques is mandatory. Full AMG and block preconditioners have been reported as a good alternative in this scenario ([Pennacchio and Simoncini, 2009], [Plank et al., 2007]). Our results show that AMG, BD, and LDU keep the iteration count below 30 iterations no matter the level of refinement of the mesh.

Full AMG performs well when applied to the very low and low resolution meshes. However, memory issues arise when it is applied to the high resolution mesh and the simulation crashes. The large memory overhead of the method is also mentioned in [Plank et al., 2007]. The iteration count for BD is around 4 iterations higher than full AMG. However, its lower cost per iteration makes it quicker in absolute terms. Furthermore, BD works with the high resolution mesh. LDU performs well for the low and high resolution meshes, being quicker than BD in both cases. However, for the very low resolution mesh its iteration count is 50% higher that BD. This behaviour is consistent with Figure 5.6, where it can be seen that convergence improves as the mesh is refined.

The dependence of the optimal preconditioning strategy on the stimulus protocol can be better understood if we look separately at the part of the residual associated with each of the magnitudes of interest: V and ϕ_e. If we rewrite (3.93) as

$$A \begin{pmatrix} \mathbf{x}_V \\ \mathbf{x}_{\phi_e} \end{pmatrix} = \begin{pmatrix} \mathbf{b}_V \\ 0 \end{pmatrix} \tag{5.4}$$

the residual in the i-th iteration of PCG can be split into its V and ϕ_e components

$$\mathbf{r}^i = \begin{pmatrix} \mathbf{r}_V^i \\ \mathbf{r}_{\phi_e}^i \end{pmatrix} = \begin{pmatrix} \mathbf{b}_V \\ 0 \end{pmatrix} - A \begin{pmatrix} \mathbf{x}_V^i \\ \mathbf{x}_{\phi_e}^i \end{pmatrix} \tag{5.5}$$

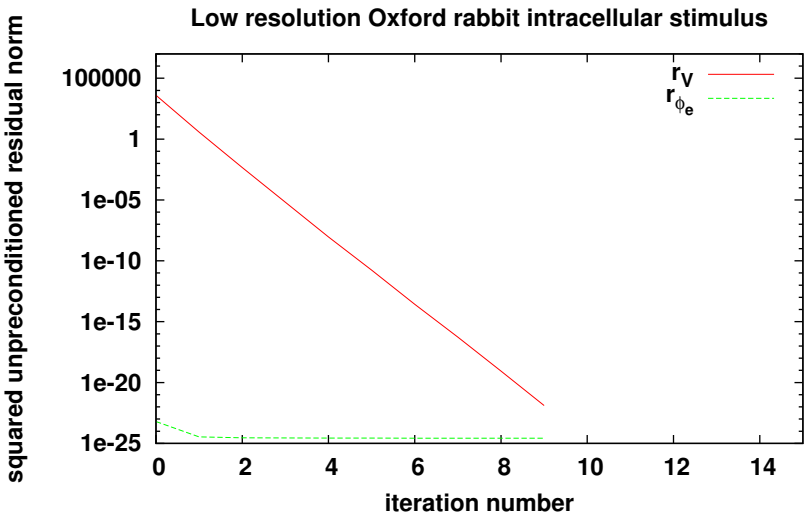
satisfying

$$||\mathbf{r}^i||^2 = ||\mathbf{r}_V^i||^2 + ||\mathbf{r}_{\phi_e}^i||^2 \,. \tag{5.6}$$

Figure 5.7 shows how in the absence of an extracellular shock, $||\mathbf{r}_{\phi_e}||$ is almost negligible.

However, the application of shocks induces sudden changes in ϕ_e across the domain. This translates into a considerable increase in $||\mathbf{r}_{\phi_e}^0||$ as shown in Figure 5.8.

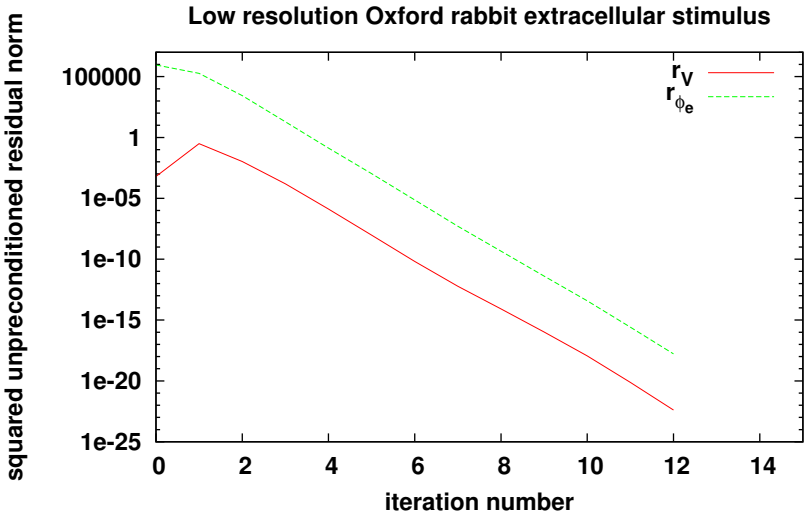
Figure 5.7 Evolution of the residual associated with each of the magnitudes of interest (V and ϕ_e) at onset of stimulus. Simulation described in Table 5.2-A with low resolution mesh without bath and PE-ILU solver.



In this scenario, ILU(0) converges to the solution very slowly and more sophisticated techniques such as LDU reduce simulation time by an important factor. This case is of particular relevance to us because our application of interest — shock-induced arrhythmogenesis — requires the simulation of such shocks.

In this chapter, we have only considered the sequential solution of bidomain linear systems. However, we saw in the Introduction that the size and complexity of the geometrical models under study and the volume of simulations to be performed in order to characterise response to different shock configurations makes the use of fully parallelised bidomain solvers mandatory. The next chapter describes the parallelisation of the techniques presented in the current chapter and their evaluation in massively parallel architectures.

Figure 5.8 Evolution of the residual associated with each of the magnitudes of interest (V and ϕ_e) at onset of stimulus. Simulation described in Table 5.2-B with low resolution mesh with bath and PE-LDU solver.



Chapter 6

Efficient parallel preconditioning for bidomain linear systems

In Chapter 4, we showed that the solution of the linear system arising from the FEM discretisation of the bidomain equations is the main bottleneck in terms of performance and parallel scalability in simulations with state-of-the-art ventricular models. In Chapter 5, we presented mesh-independent bidomain preconditioners and showed that, for our problem configuration of interest, they substantially reduce execution time when compared to generic preconditioning techniques. However, these techniques were only implemented and evaluated sequentially. As we justified in the Introduction, the computational requirements of our application of interest make the use of parallel hardware mandatory.

In this chapter we present and evaluate parallel implementations of the two problem-specific block preconditioners (BD and LDU) introduced in Chapter 5. Both rely upon providing a reliable and practical method for calculating the action of the inverse of certain matrix subblocks. Results show that: a) total execution time can be reduced by using different inverse approximation methods for different matrix subblocks, b) there exists a good correlation between the coefficient $\Delta t/h^2$ (i.e. PDE solution timestep over maximum grid edge length squared) and the cases

where BD and LDU outperform generic Algebraic Multigrid (AMG) preconditioning, and c) when applied to simulations with current state-of-the-art cardiac geometries, BD and LDU are over 3 times faster than AMG preconditioning and show similar scaling. The results in this chapter have been published in [Bernabeu and Kay, 2011].

6.1 Introduction

In Chapter 5, we presented two efficient sequential preconditioners for the solution of the bidomain equations on unstructured grids, namely BD and LDU. Both achieved good performance and in most cases outperformed generic preconditioning techniques such as Algebraic Multigrid (AMG). We also proved theoretically and empirically the mesh-independent convergence properties of LDU.

In the current chapter, we will present parallel implementations of BD and LDU and extend the convergence analysis presented in Chapter 5 to other problem characteristics like the PDE solution timestep (i.e. Δt). To the best of our knowledge, no previous work has studied the influence of this parameter on the optimal choice of bidomain preconditioner.

Regarding performance, BD and LDU rely upon providing a reliable and practical method for calculating the action of the inverse of certain matrix subblocks. In Chapter 5, we used 1 AMG V-cycle, but in this work we will investigate other options. Unlike previous publications [Bernabeu et al., 2010a; Pennacchio and Simoncini, 2009; Plank et al., 2007], we will not consider AMG a black box, instead we will explore some of the configuration options available in HYPRE's BoomerAMG algorithm (e.g. coarsening and interpolation algorithms) (see Sections 3.2.3 and 3.2.4 for details of the algorithm and the software package implementing it).

Previous studies of bidomain preconditioning scalability did not go beyond a few

hundred processors. In [Pavarino and Scacchi, 2008], close to optimal weak scaling[1] is achieved with up to 240 processors in idealised ventricular domains with an Additive Schwarz preconditioner (see Section 3.2.3 for a description of the method). In [Scacchi and Pavarino, 2008], block Jacobi, geometric multigrid, and Schwarz methods (see Section 3.2.3 for a description of the methods) are compared in parallel. Performance of block Jacobi is reported to deteriorate with increasing number of processors due to an increase in iteration count. Geometric multigrid and Schwarz achieve good strong scaling with up to 16 processors and good weak scaling with up to 128. Parallel geometric multigrid and block Jacobi are also compared in [dos Santos et al., 2004] with similar results. In [Plank et al., 2007], block Jacobi and algebraic multigrid are compared with up to 64 processors. Algebraic multigrid outperforms block Jacobi for all the processor configurations considered, achieving close to optimal speedup with up to 16 processors. The authors highlight the fact that algebraic multigrid is much more flexible than its geometric counterpart since it does not require multiple discretisations of the domain to be generated and stored in memory. In the current chapter, we will present results for core counts one order of magnitude larger than any of the works cited above.

The purpose of this chapter is two-fold. First, present scalable parallel implementations of the preconditioners presented in Chapter 5. Second, study their efficiency as a function of number of processors, spatial, and temporal discretisation in simplified geometries as well as in realistic 3D anatomically-based geometries.

The preconditioners considered in this work were introduced in Section 5.2. Their parallelisation is described in Section 6.2. Section 6.3 presents the benchmarks used to evaluate performance and scalability. Finally, Section 6.4 presents the benchmarks results and Section 6.5 analyses them and summarises the main findings.

[1]Recall, in the context of high performance computing there are two different notions of scalability. The first is strong scaling, which quantifies how the solution time varies with number of processors for a fixed problem size. The second is weak scaling, which quantifies how the solution time varies with number of processors for a fixed problem size per processor.

6.2 Preconditioner parallelisation and tuning

Recall the block structure of (3.104):

$$A\mathbf{x} = \begin{bmatrix} A_1 & B^T \\ B & A_2 \end{bmatrix} \mathbf{x} = \mathbf{b} \tag{6.1}$$

with blocks

$$A_1 := \begin{bmatrix} \frac{\chi\mathcal{C}}{\Delta t} M + K[\sigma_i] & 0 \\ 0 & I_M \end{bmatrix}, \quad B := \begin{bmatrix} K[\sigma_i] & 0 \\ 0 & 0 \end{bmatrix} \quad \text{and } A_2 := \mathcal{K}. \tag{6.2}$$

where $A \in \mathbb{R}^{2(N+M)\times 2(N+M)}$, $\mathbf{x}, \mathbf{b} \in \mathbb{R}^{2(N+M)}$, $A_1, A_2, B \in \mathbb{R}^{(N+M)\times(N+M)}$, and $I_M \in \mathbb{R}^{M\times M}$ is an identity matrix.

In every CG iteration $i+1$ (see Section 3.2.1 for a description of the method), the following preconditioning operation is performed:

$$\mathbf{z}^{(i)} = P^{-1}\mathbf{r}^{(i)}\,, \tag{6.3}$$

where $\mathbf{r}^{(i)}$ is the residual of the solution in the previous timestep, P^{-1} is the preconditioner and $\mathbf{z}^{(i)}$ is used in the computation of the next search direction. Note that depending on the choice of preconditioner this may or may not involve an explicit matrix-vector product but the application of a linear operator. In this section, we will consider $P^{-1} = \mathcal{P}_{BD}^{-1}$ but the discussion can be easily extended to $\mathcal{P}_{LDU}^{-1}$. For performance reasons, and assuming that $\mathbf{r}^{(i)} = (\mathbf{r}_V^{(i)}, \mathbf{r}_{\phi_e}^{(i)})^T, \mathbf{r}_{\{V,\phi_e\}}^{(i)} \in \mathbb{R}^{N+M}$ and similarly with $\mathbf{z}^{(i)}$, (6.3) can be rewritten as

$$\mathbf{z}_V^{(i)} = A_1^{-1}\mathbf{r}_V^{(i)} \tag{6.4}$$

$$\mathbf{z}_{\phi_e}^{(i)} = A_2^{-1}\mathbf{r}_{\phi_e}^{(i)} \tag{6.5}$$

In order to ensure good parallel scalability in the application of $\mathcal{P}_{BD}^{-1}$ one has to minimise the amount of data to be communicated between processors for the assembly of $\mathbf{r}_V^{(i)}$ and $\mathbf{r}_{\phi_e}^{(i)}$ from $\mathbf{r}^{(i)}$ (scatter operation) and $\mathbf{z}^{(i)}$ from $\mathbf{z}_V^{(i)}$ and $\mathbf{z}_{\phi_e}^{(i)}$ (gather operation) at each timestep. We will now show how our underlying data structures ensure this. If we have a closer look at how (6.3), (6.4), and (6.5) are distributed (based on the data layout of model problem (4.13), where the solid line represents how vectors are distributed between processors.)

$$\left(\begin{array}{c} \mathbf{z}_{V_1}^{(i)} \\ \hline \mathbf{z}_{\phi_1}^{(i)} \\ \hline \mathbf{z}_{V_2}^{(i)} \\ \mathbf{z}_{\phi_2}^{(i)} \end{array}\right) = M^{-1} \left(\begin{array}{c} \mathbf{r}_{V_1}^{(i)} \\ \hline \mathbf{r}_{\phi_1}^{(i)} \\ \hline \mathbf{r}_{V_2}^{(i)} \\ \mathbf{r}_{\phi_2}^{(i)} \end{array}\right), \qquad \left(\begin{array}{c} \mathbf{z}_{V_1}^{(i)} \\ \hline \mathbf{z}_{V_2}^{(i)} \end{array}\right) = A_1^{-1} \left(\begin{array}{c} \mathbf{r}_{V_1}^{(i)} \\ \hline \mathbf{r}_{V_2}^{(i)} \end{array}\right), \qquad \left(\begin{array}{c} \mathbf{z}_{\phi_1}^{(i)} \\ \hline \mathbf{z}_{\phi_2}^{(i)} \end{array}\right) = A_2^{-1} \left(\begin{array}{c} \mathbf{r}_{\phi_1}^{(i)} \\ \hline \mathbf{r}_{\phi_2}^{(i)} \end{array}\right)$$

it can be seen that data structures $\mathbf{r}_V^{(i)}$, $\mathbf{r}_{\phi_e}^{(i)}$ and $\mathbf{z}^{(i)}$ can be assembled with data local to each processor, therefore avoiding communication.

If we were not using the interleaved data layout described in Section 4.4, $\mathbf{r}_V$ would be entirely owned by the processors owning the first $M + N$ equations and $\mathbf{r}_\phi$ by the rest. One could think that (6.4) and (6.5) can be executed independently and therefore data does not need to be interchanged with the other half of the processors. Later we will see how different inverse approximations are used for (6.4) and (6.5) yielding different computational cost. Therefore, decoupling their computation would compromise load balancing. For $\mathcal{P}_{LDU}^{-1}$ the situation is even worse because (6.4) needs to be solved twice per CG iteration.

6.3 Benchmarks

In this section we describe the benchmarks designed to evaluate performance and scalability of the parallel implementations of BD and LDU developed in this chapter.

6.3.1 Preconditioners tuning

In Chapter 5, we used one AMG V-cycle for the approximation of A_1^{-1} and A_2^{-1} in BD and LDU. We would like to study alternatives for the approximation of A_1^{-1} and A_2^{-1}. This is motivated by the fact that factorisation-based approximations are often faster per iteration taken. We also want to investigate domain decomposition techniques such as Block Jacobi (BJ) or Additive Schwarz Method (ASM) for their benefits in terms of communication reduction (see Section 3.2.3 for a description of the methods). Multiple combinations of these techniques are compared for the solution of a benchmark consisting of 2 ms of bidomain simulation on the low resolution version of the Oxford rabbit mesh (see Section 2.3 for a description of the geometry). Tissue is stimulated with a 0.5 ms long extracellular shock of magnitude $-11 \cdot 10^3$ μA/cm^2 delivered at $t = 0$ ms through electrode plates located at the boundaries of the bath parallel to the sagittal plane. The cell model used is [Luo and Rudy, 1991] integrated with a backward Euler numerical scheme. Both PDE and ODE timesteps are 0.01 ms.

6.3.2 Δt- and h-dependence analysis

We will extend the mesh-dependence analysis presented in Chapter 5 by evaluating how Δt affects convergence of the preconditioners presented in Section 3.2.3. For this purpose, we generated a set of five 3D tissue slabs immersed in conductive bath. The original volume is a $11 \times 11 \times 11$ mm cube meshed as a regular grid with constant distance between nodes along each dimension $h \in \{1, 0.5, 0.25, 0.125, 0.0625\}$ mm. Nodes in the internal $7 \times 7 \times 7$ mm volume are defined as tissue and the rest as bath. Tissue is stimulated with a 1 ms long extracellular shock of magnitude $-11 \cdot 10^3$ μA/cm^2 delivered at $t = 0$ ms through electrode plates located at the boundaries of the bath in the yz plane. Stimulation is applied for long enough to trigger depolarisation in the area of tissue closer to the negative electrode. After

that, electrodes are switched off and a planar wave travels along the x direction. The cell model used is [Luo and Rudy, 1991], integrated with a backward Euler numerical scheme. PDE timesteps considered are 0.1, 0.05, and 0.01 ms. ODE timestep is 0.01 ms.

In order to also study parallel scaling of the preconditioners, the following combinations of number of processors and discretisations were considered: i) $p = 16$, $h \in \{1, 0.5, 0.25\}$ mm, ii) $p = 128$, $h \in \{0.5, 0.25, 0.125\}$ mm, and iii) $p = 1024$, $h \in \{0.25, 0.125, 0.0625\}$ mm. This choice ensures that the number of nodes per processor stays almost constant for the three experiments run no matter the value of p.

6.3.3 Realistic 3D simulations strong scaling

In order to evaluate the techniques presented in realistic 3D cardiac models, we designed a benchmark with the Oxford rabbit mesh. Bidomain activity is simulated for 5 ms. In order to elicit propagation, a 0.5 ms long square extracellular shock of magnitude $-11 \cdot 10^3$ μA/cm^2 is delivered at $t = 0.2$ ms through electrode plates located at the boundaries of the bath parallel to the saggittal plane. The cell model used is [Luo and Rudy, 1991], integrated with a backward Euler numerical scheme. PDE timestep is 0.1 ms and ODE timestep is 0.01 ms.

6.4 Results

All the simulations presented in this chapter were run with Chaste. The following parameters were used in equations (3.95)–(3.99): $\chi = 1400$ cm^{-1}, $\mathcal{C} = 1.0$ μF/cm^2, $\sigma_b = 8.0$ mS/cm, $\sigma_i = diag(1.7, 0.19, 0.19)$ mS/cm, and $\sigma_e = diag(6.2, 2.4, 2.4)$ mS/cm, where $diag(x, y, z)$ is a 3×3 diagonal matrix with values x, y, z along the diagonal. Parallel AMG preconditioning is performed with the BoomerAMG algorithm found

in the HYPRE library. We choose this AMG implementation for its maturity in terms of parallel performance [Baker et al., 2010] and for having been successfully applied to the solution of the bidomain equations before [Bernabeu et al., 2010a; Plank et al., 2007]. BoomerAMG is a sophisticated algorithm with a large number of configuration options that can have a real impact on performance. A complete exploration of the parameter configuration space is beyond the scope of this work. However, we tuned the parameters considered to have the greatest impact in parallel performance: strong connectivity threshold, coarsening algorithm, smoother and interpolation algorithm.

The impact of strong connectivity threshold in BoomerAMG performance for the solution of the bidomain equations has been studied before. Therefore, we use the value of 0.0 reported as optimal in [Plank et al., 2007]. From the available smoothers, we chose Hybrid Gauss-Seidel based on the scalability results presented in [Baker et al., 2010]. In that work, the authors prove that Hybrid Gauss-Seidel scales well provided that the relative weight of the off-diagonal block of a row-based partitioned system matrix, θ, is strictly larger than 1 [Baker et al., 2010, Definition (5.3)] (not to be mistaken with the strong connectivity threshold also referred as θ in [Plank et al., 2007]). Several coarsening algorithms are compared in Section 6.4.1. The interpolation algorithm is set to BoomerAMG's default: `classical`, since experiments with alternative configurations did not yield any improvement.

Left preconditioning is used throughout this work. In order to make our comparisons as fair as possible, we configure PETSc to use unpreconditioned residual for convergence testing. This is known to introduce an overhead when evaluating the convergence condition. A relative tolerance of 10^{-10} and PETSc's default stop criteria is used. Preconditioners and benchmarks are available in Chaste release 2.2 and later.

All the simulations presented were run in HECToR Phase 2a. This is a Cray

XT4 system with 3072 compute nodes (at the time of writing). Each compute node is a AMD 2.3 GHz Opteron Barcelona quad-core. Each quad-core socket shares 8 GB of memory and a Cray SeaStar2 chip router with 6 links which are used to implement a 3D-torus network topology.

6.4.1 Preconditioners tuning

Table 6.1 compares different choices of block inverse approximation for the benchmark described in Section 6.3.1 run with 8 processors.

Table 6.1 Total linear system solution time and average number of iterations for benchmark described in Section 6.3.1. BJ(xxx) means BJ domain decomposition with xxx preconditioning at each block and similarly with ASM(xxx). AMG(yyy) means 1 AMG V-cycle with yyy coarsening.

A_1^{-1} approximation	A_2^{-1} approximation	CG time (s)	average iterations
AMG(Falgout)	AMG(Falgout)	651	15.23
BJ(ILU(0))	BJ(ILU(0))	–	aborted after 200
BJ(ILU(0))	BJ(AMG(Falgout))	2340	63.48
BJ(ILU(0))	ASM(AMG(Falgout))	4230	30.13
BJ(ILU(0))	AMG(Falgout)	518	11.58
BJ(ILU(0))	AMG(Ruge-Stueben)	498	11.25
BJ(ILU(0))	AMG(PMIS)	464	21.55
BJ(ILU(0))	AMG(HMIS or CLIP)	431	19.37
ILU(0)	AMG (HMIS)	406	17.46

6.4.2 Δt-/h-dependence

Table 6.2, 6.3, and 6.4 report average number of iterations per solve and total linear system solve time (over the whole simulation) for the benchmark described in Section 6.3.2.

6.4.3 Strong scaling

Table 6.5 reports average number of iterations per solve and total linear system solve time (over 50 solves) for the benchmark described in Section 6.3.3. Figure 6.1 plots

Table 6.2 Δt-/h-dependence benchmark with $p = 16$. (a) average number of iterations, (b) linear solve execution time (s)

Δt	h			preconditioner
	1 mm	0.5 mm	0.25 mm	
0.1 ms	12.07	13.3	14.72	AMG
	13.18	16.15	19.51	BD
	12.14	13.97	16.17	LDU
0.05 ms	11.42	12.37	13.64	AMG
	12.01	14.63	17.57	BD
	11.4	13.07	14.55	LDU
0.01 ms	9.93	10.65	11.38	AMG
	10.13	11.79	13.66	BD
	9.85	11.29	12.31	LDU

(a)

Δt	h			preconditioner
	1 mm	0.5 mm	0.25 mm	
0.1 ms	4.19	8.24	38.26	AMG
	4.61	8.57	30.37	BD
	4.76	8.38	28.44	LDU
0.05 ms	7.6	15.34	71.03	AMG
	8.4	15.54	54.88	BD
	8.96	15.66	51.21	LDU
0.01 ms	33.1	66.19	297.99	AMG
	35.48	62.71	215.89	BD
	38.7	67.62	219.44	LDU

(b)

Table 6.3 Δt-/h-dependence benchmark with $p = 128$. (a) average number of iterations, (b) linear solve execution time (s)

Δt	h			preconditioner
	0.5 mm	0.25 mm	0.125 mm	
0.1 ms	14.93	16.3	34.84	AMG
	17.63	21.67	23.92	BD
	15.23	18.06	20.76	LDU
0.05 ms	13.92	15.65	27.88	AMG
	15.83	19.4	21.63	BD
	14.26	16.34	18.23	LDU
0.01 ms	12.36	12.99	13.37	AMG
	12.67	15.14	16.94	BD
	12.28	13.79	14.17	LDU

(a)

Δt	h			preconditioner
	0.5 mm	0.25 mm	0.125 mm	
0.1 ms	13.02	17.89	161.65	AMG
	14.94	22.62	52.85	BD
	14.34	20.94	52.77	LDU
0.05 ms	22.07	34.48	252.63	AMG
	26.8	40.78	95.94	BD
	26.79	38.02	92.63	LDU
0.01 ms	98.78	143.91	471.04	AMG
	108.03	164.7	445.46	BD
	115.65	171.84	426.52	LDU

(b)

Table 6.4 Δt-/h-dependence benchmark with $p = 1024$. (a) average number of iterations, (b) linear solve execution time (s)

Δt	h			preconditioner
	0.25 mm	0.125 mm	0.0625 mm	
0.1 ms	18.73	35.43	61.18	AMG
	23.45	26	30.79	BD
	19.78	22.78	27.98	LDU
0.05 ms	17.43	28.42	57.97	AMG
	21.19	23.44	27.63	BD
	18	20.17	24.62	LDU
0.01 ms	14.84	15.57	40.77	AMG
	16.6	18.39	21.42	BD
	15.25	15.72	18.33	LDU

(a)

Δt	h			preconditioner
	0.25 mm	0.125 mm	0.0625 mm	
0.1 ms	24.03	64.68	304.02	AMG
	31.15	44.12	90.28	BD
	28.75	40.96	92.87	LDU
0.05 ms	44.7	102.9	576.27	AMG
	56	77.24	159.71	BD
	52.67	72.47	161.99	LDU
0.01 ms	199.24	279.1	2114.35	AMG
	245.55	336.44	670.58	BD
	247.05	316.19	652.76	LDU

(b)

parallel scalability.

Table 6.5 Realistic 3D simulation benchmark. (a) Average CG iterations, (b) Total CG time (s)

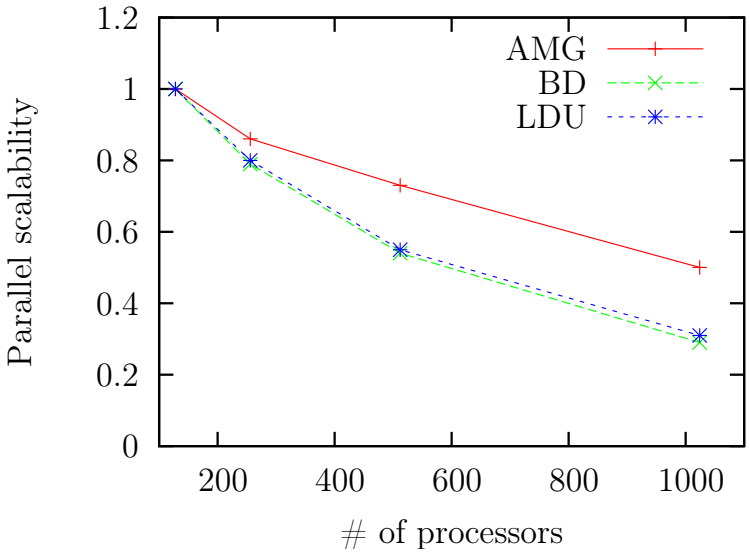
p	AMG	BD	LDU
128	109.14	39.92	36.32
256	109.5	39.86	36.2
512	110.22	40.58	37.06
1024	110.62	42.86	39.2

(a)

p	AMG (s)	BD (s)	LDU (s)
128	1310.36	221.48	240.58
256	762.62	140.04	150.46
512	447.43	102.37	109.29
1024	328.1	93.99	97.68

(b)

Figure 6.1 Realistic 3D simulation benchmark strong scaling comparison. Scalability is defined with respect to $p = 128$.



6.5 Discussion and conclusions

In this chapter, we presented and evaluated parallel implementations of two preconditioners proposed in Chapter 5. In sequential, both show better performance than generic preconditioning techniques such as ILU(0) or AMG. However, their parallelisation is challenging because they potentially require data migration for the

assembly of intermediate data structures, which can compromise their scalability. Our implementation uses a data partitioning scheme that ensures no data migration when assembling these intermediate data structures. This guarantees that parallel scalability is only restricted by the underlying inverse approximation algorithms and therefore future scalability improvements to them will automatically improve BD and LDU without need of recoding.

In Chapter 5, we considered 1 AMG V-cycle for the approximation of the action of A_1^{-1} and A_2^{-1} with HYPRE's BoomerAMG algorithm default configuration. In this work, we investigated other options. Table 6.1 shows how using ILU(0) or block Jacobi with ILU(0) for the approximation of A_1^{-1} slightly increases the average number of iterations but reduces total execution time. This is not the case with A_2^{-1}, where the iteration count increases dramatically with approximations different from 1 AMG V-cycle. Table 6.1 also shows how the choice of coarsening algorithm has an important impact on performance. In summary, we reduced by 38% the total linear system solution time of the benchmark presented in Section 6.3.1 by using ILU(0) for A_1^{-1} approximation and 1 AMG V-cycle with HMIS coarsening for A_2^{-1} compared with the original setup (1 AMG V-cycle with Falgout coarsening for both approximations). The downside to this improvement is that LDU loses the mesh-independent convergence properties proved in Chapter 5. Therefore, we expect that there will be a cutoff value of h below which approximating A_1^{-1} with AMG will reduce overall execution time. This has been investigated and for the finest discretisation considered in this work $h = 0.0625$ mm (which is twice as fine as current state-of-the-art cardiac models) approximating A_1^{-1} with ILU(0) is still between 19% and 43% faster (depending on the choice of Δt) than using 1 AMG V-cycle.

We also extended our original h-dependence analysis with a combined h-/Δt-dependence analysis. Results show a good correlation between large values of $\Delta t/h^2$

and the situations where BD and LDU outperform AMG. This is due to the fact that the condition number of $\mathcal{P}^{-1}\mathcal{A}$ is a function of $\Delta t/h^2$ when mesh-dependent preconditioners are considered [Wathen, 1987]. Therefore, the number of iterations required by AMG will grow unbounded as the condition number increases. In contrast, LDU iteration count either: i) converges to an asymptotic bound when 1 AMG V-cycle is used for the approximation of both subblocks, or ii) grows at a smaller rate when the efficacy of the first block inverse approximation is relaxed (see Table 6.4). Table 6.6 gives the value of the coefficient for all the combinations of Δt and h considered and highlights in dark grey cases where BD and LDU outperform AMG. In our experiments, this happens for values of $\Delta t/h^2$ greater than 2.5. For values below 2.5, the system is not so ill-conditioned and all the preconditioners considered keep iteration count low. In that scenario, the slightly lower computational cost per iteration of AMG makes it faster than BD and LDU.

Table 6.6 Coefficient $\Delta t/h^2$ for all the combinations of Δt (ms) and h (mm) considered. In dark grey, cases where BD and LDU outperform AMG for any number of processors. In light grey, configurations where this only happens for low core counts

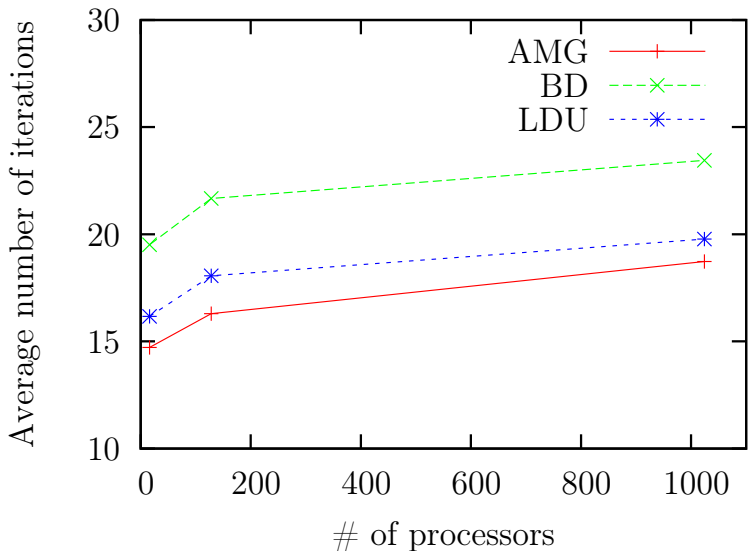
Δt \ h	1	0.5	0.25	0.125	0.0625
0.1	0.1	0.4	1.6	6.4	25.6
0.05	0.05	0.2	0.8	3.2	12.8
0.01	0.01	0.04	0.16	0.64	2.56

Interesting exceptions are the combinations highlighted in light grey in Table 6.6 and in particular $h = 0.25$ mm, $p = 16$ (Table 6.2) where AMG has a lower iteration count than LDU, but the latter is around 27% faster than the former. However, this result does not hold for $p = 128$ (Table 6.3) and $p = 1024$ (Table 6.4) as one would expect for a value of $\Delta t/h^2$ below the given bound. We believe that in this case the number of degrees of freedom per processor also has an impact on performance. More precisely, in scenarios where a large number of degrees of freedom are stored in each processor, the fact that LDU (and BD) works with two subproblems half

the size of the one considered by AMG may have a positive impact in memory utilisation reducing cache misses and/or page fault rates, effectively shifting the empirical bound given before. Further profiling needs to be carried out to confirm this hypothesis, though.

In agreement with [Baker et al., 2010], we see a slight increase in the average iteration count with the number of processors for any fixed problem. This is due to a decrease in the relative weight of the off-diagonal block θ. Figure 6.2 plots average number of iterations for $h = 0.25$ mm, $\Delta t = 0.1$ ms and 16, 128, and 1024 processors. Similar results are observed in the realistic 3D benchmark.

Figure 6.2 Average number of iterations for benchmark described in Section 6.3.2, $h = 0.25$ mm, and $\Delta t = 0.1$ ms



Finally, strong scaling of the three methods (Figure 6.1) is good although not excellent. The reasons are: i) the increase in iteration count with number of processors described above, and ii) the tightly-coupled nature of the approximation algorithms. Furthermore, we see that strong scaling is slightly better for AMG than for BD and LDU. This is due to the fact that in AMG a single problem of size $2M + 2N$ is considered while in BD and LDU two subproblems of $M + N$ are solved. This reduces the number of degrees of freedom per processor for a fixed problem size and

therefore scalability tails off faster. Nevertheless, total linear system solution time with BD and LDU is still over 3 times faster than with AMG for the range of core counts considered in the strong scaling study.

Chapters 4–6 have presented the developments undertaken to bring Chaste to the level of scalability and performance required for the in-silico study of defibrillation processes in the human heart. This includes efficient numerical methods and scalable implementations. We can now run bidomain simulations with highly-detailed ventricular models including representation of the surrounding media in time frames that allow the exploration of parameter spaces in order to charaterise response to electrical shocks. In the next Chapter, we will firstly develop a human model of cardiac defibrillation including automatic definition of tissue heterogeneities and we will secondly perform the first in-silico study of shock-induced arrhythmogenesis in the human heart.

Chapter 7

Shock-induced arrhythmogenesis in the human heart: a simulation study

Electrical defibrillation through the application of electric shocks to the heart is the only effective therapy against ventricular fibrillation. However, factors underlying the success or failure of defibrillation shocks remain unclear. Over the past years, defibrillation mechanisms have been extensively investigated on small animal models such as the rabbit (see Section 2.2.3 for a survey). Nevertheless, authors [Maharaj et al., 2008; Rodríguez et al., 2004] often acknowledge that extrapolation of their results to the human has to be made with caution due to differences in size, anatomy, and electrophysiology. Of particular relevance is the increase in wall thickness and the decrease of the proportion of tissue under the effects of tissue-bath interactions.

The main limitation to conduct defibrillation studies using human ventricular models is the lack of the appropriate simulation technology. Challenges are associated with the need of running a large number of bidomain simulations on spatially detailed geometries made of tens of millions of grid points and with precise description of fibre structure and membrane kinetics. Performance limitations were

identified and addressed in Chapters 4–6 of this Thesis and solutions were implemented in the framework of the open source package Chaste.

The purpose of this chapter is therefore two-fold: i) to develop a human model of cardiac defibrillation with patient-specific anatomy including a generic framework for fibre orientation generation and ii) to perform the first simulation study of shock-induced arrhythmogenesis in the human heart. The objective of this study is to understand how geometrical differences between human and rabbit hearts affect shock-induced electrical activity. We hypothesise that differences in wall thickness have an impact in the importance of tissue-bath interaction in shock-end polarisation and post-shock behaviour.

The rest of the chapter is structured as follows: Section 7.1 describes the geometry used in our study, Section 7.2 presents the framework for automatic generation of tissue heterogeneities implemented in Chaste, and finally Section 7.3 reports the results of the study described above.

7.1 Human ventricular anatomy

In this work we use a patient-specific anatomical model developed by Mikael Wallman [Bernabeu et al., 2010b]. The model includes representation of both ventricles of a human heart. The model was generated from a ventricular surface definition obtained from computed tomography (CT) images. Data was originally provided by the CISTIB group of the Pompeu Fabra University, Spain.

Starting from a surface made up of triangular elements, a semi-automatic algorithm was developed to generate an image stack, representing voxels. The voxel data was subsequently used to generate a tetrahedral mesh using Tarantula[1]. The algorithm consists of four steps. First, the surface mesh is rotated so that the cen-

[1]Tarantula is a meshing tool for voxel-data coming from medical imaging techniques (e.g. MRT/CT). Tarantula is a comercial tool developed by CAE-Software Solutions, `www.meshing.at`.

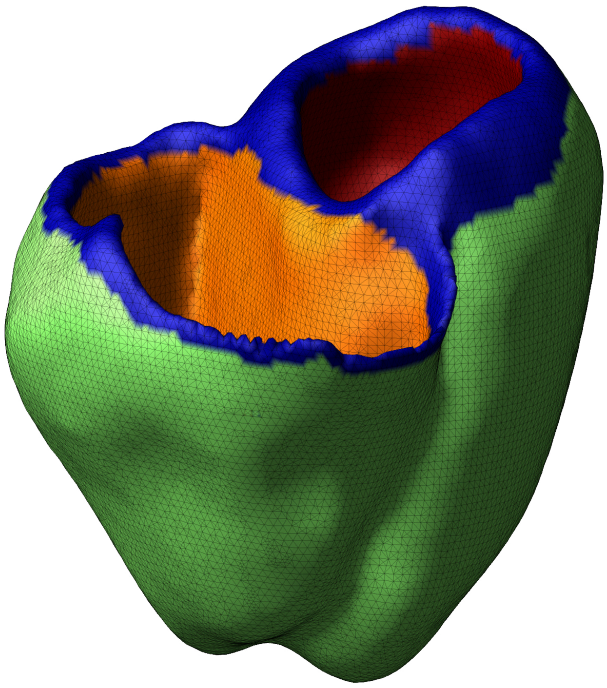
tral axis of the geometry is approximately oriented in the z-direction. Second, the intersections of the surface with a stack of planes, each of which corresponded to one of the images of the resulting image stack, are calculated. Third, the closed curves resulting from each intersection are classified as being empty or filled, and fourth, the images of the image stack are generated based on this classification. The algorithm takes the maximum edge length in the tetrahedral mesh as an input parameter. The version of the mesh used in the simulation study presented in the next section has an maximum edge length of 400 μm and consists of 5.2M nodes and 30.9M elements, including representation of tissue, blood in the ventricular cavities, and surrounding media. A semi-automatic algorithm was developed to classify the epicardial surface enveloping both ventricles and the endocardial surface of the left and right ventricles separately. Figure 7.1 shows an anterior view of the model with the different surfaces colour coded. The reader can refer to [Bernabeu et al., 2010b] for a detailed description of both algorithms.

7.2 Automatic definition of tissue heterogeneity properties

In Chapter 2, we saw that myocytes are arranged in bundles. These bundles branch and interconnect but collagen planes are extensive among them with no myocyte to myocyte coupling happening across the collagen. Electrical conductivity is, therefore, anisotropic.

Myocardial fibre orientation is known to be crucial for sustainability of the sinus rhythm and also in the response to electrical shocks [Hooks et al., 2002; Rodríguez et al., 2005; Trayanova et al., 2006; Trew et al., 2009]. Therefore, it needs to be taken into account for the development of realistic in-silico models of the cardiac defibrillation. Fibre orientation — at least in the ventricles — can be defined as

Figure 7.1 Human ventricular mesh generated from CT images including surface classification. Epicardium is shown in green, left ventricle in red, right ventricle in orange and border zone in blue. Elements defining the content of the ventricular cavities and the surrounding media are not plotted. Image courtesy of Mikael Wallman.



a function of the position of a given cell within the myocardium [Streeter et al., 1969]. In this section, we develop a generic framework for definition of anisotropy properties based on the relative position of the mesh nodes within the ventricular wall according to the model presented in [Potse et al., 2006]. The work in this section is developed as a preliminary step before conducting the simulation study presented in the next section.

7.2.1 An algorithm for distance map calculation in cardiac geometries

The automatic definition of fibre orientation based on the observations of [Streeter et al., 1969] requires knowledge about the relative position of a node within the heart. Ideally, this information can be extracted from the medical images originally used to develop the geometrical model. In some cases this is not possible and specialised algorithms need to be developed to compute the distance between a node and certain surfaces of interest (e.g. epicardium, endorcardium). In the context of image processing, a similar problem was formalised in [Rosenfeld and Pfaltz, 1966]: the distance transform[2] (DT) maps each of the pixels of an image into its smallest distance to a region of interest. It is a fundamental geometrical operator with great applicability in computer vision and graphics, shape analysis, pattern recognition, and computational geometry.

The DT can be defined in terms of arbitrary metrics. The Euclidean distance between two points $\mathbf{x}, \mathbf{y} \in \mathbb{R}^k$

$$d(\mathbf{x}, \mathbf{y}) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \ldots + (x_k - y_k)^2}$$

is often necessary in many applications, as it is the adequate model to numerous

[2]Also known as distance map or distance field.

geometrical facts of the real world. Other metrics can be used to model the distance between $\mathbf{x}$ and $\mathbf{y}$. Examples are the *city-block* or *manhattan* metric

$$d_1(\mathbf{x}, \mathbf{y}) = |x_1 - y_1| + |x_2 - y_2| + \ldots + |x_k - y_k|$$

or the *chessboard* metric

$$d_\infty(\mathbf{x}, \mathbf{y}) = max\{|x_1 - y_1|, |x_2 - y_2|, \ldots, |x_k - y_k|\}$$

These metrics are sometimes preferred because they are computationally less expensive than the Euclidean metric. Unfortunately, they do not have some of the properties required by our application, such as being radially symmetric. For the rest of the section, we will restrict our results to Euclidean distance transforms (EDT)

The central problem of a DT is to compute the distance from each point of a k-dimensional space to a given subset of it (i.e. DT origin). Let $M = (N, E)$ be a computational mesh, where N and E are the set of nodes and elements respectively. Let $S \subset N$ be the subset of nodes that define the origin of the DT. The DT is the transformation that generates a map D whose value in each node $n \in N$ is the smallest distance from n to S:

$$D(n) = min\{d(n, s) | s \in S\}$$

with

$$D(s) = 0 \ \ \forall s \in S$$

D will be referred as the *distance map* of $\langle M, S \rangle$.

A brute force approach for the solution of this problem in a convex region would be to compute the distance between each point in N and all the points in S and

take the minimum. It is straight-forward to see that the temporal cost of this algorithm is a function of the size of the region of interest, $|S|$. The number of comparisons performed by such an algorithm is $|S| \times (|N| - |S|)$. This function reaches its maximum at $|S| = |N|/2$, hence the maximum number of operations is $|N|/2 \times (|N| - |N|/2) = |N|^2/4 = O(|N|^2)$[Fabbri et al., 2008]. The minimum non-zero number of operations occurs for $|S| = \{1, |N| - 1\}$. Therefore, the brute force algorithm is $\Omega(|N|)$ and only optimal for the trivial case. As every DT algorithm has to visit each node at least once, the optimal algorithm runs in $O(|N|)$. In fact, some authors have already proposed algorithms with this complexity, see [Fabbri et al., 2008] for more details.

In the context of image processing, algorithms have been developed to compute distance maps in regular 2D and 3D grids representing binary images. [Fabbri et al., 2008] and [Jones et al., 2006] present complete surveys on the most common algorithms and analyse their accuracy and performance. [Jones et al., 2006] provides an algorithm classification based on two criteria. Firstly, based on how we estimate the distance value of a given pixel/voxel from the known quantities of its neighbours:

chamfer DTs where distances are propagated between neighbours by adding scalar values,

Euclidean (or vector) DTs where each processed pixel/voxel stores a vector to its nearest point in S, and

fast marching methods where the distance of a pixel/voxel is computed by solving the Eikonal equation

$$|\nabla T| = \frac{1}{F}$$

with $T|_S = 0$. $F \geq 0$ is the speed of the front, and T is the arrival time of the front at a given point p.

Secondly, based on how distances are propagated through the computational domain:

sweeping scheme when the propagation starts in one corner of the volume and proceeds in a voxel-by-voxel, row-by-row fashion to the opposite corner, typically requiring several passes in different directions.

wavefront scheme when the distances are propagated from the initial surface in increasing distance order until all voxels are processed.

Finally, [Jones et al., 2006] concludes that wavefront propagation DT algorithms with the Euclidean metric show the best results in terms of accuracy and performance. This will be our choice for the design of a DT algorithm for cardiac geometries.

All the works cited so far present algorithms and results for 2D and 3D uniform grids naturally arising in image processing. Two works [Botsch et al., 2004; Sud et al., 2004] were found where DTs are applied to triangular meshes, but always defined over structured grids. To the best of our knowledge, no previous work has applied these techniques to unstructured grids.

Algorithms for structured grids can be optimised simply by checking against a given subset of the neighbours of a pixel/voxel. This set is known as a *directed mask* [Ragnemalm, 1992] or a *vector template* [Jones et al., 2006]. [Ragnemalm, 1992] proves that in the 2D case, distance maps can be computed using only two neighbours in most masks. Unfortunately, the properties exploited in these optimisations do not hold for unstructured grids. The algorithm proposed is conservative in that sense. When a node is being processed, all its neighbours are visited. We believe that a better understanding of the nature of our unstructured grids may allow us to skip some of them in order to increase performance.

The following algorithm is based on the work of Ragnemalm [Ragnemalm, 1992], although it can be seen as a generic propagation algorithm with vector Euclidean

metric. The operations have been adapted to match our data structures and the characteristics of our problem in general. Throughout the algorithm, two different distance metrics will be used. The vector Euclidean distance between two points $\mathbf{x}, \mathbf{y} \in \mathbb{R}^{dim}$ is defined as:

$$v(\mathbf{x}, \mathbf{y}) = (x_1 - y_1, x_2 - y_2, \ldots, x_{dim} - y_{dim})$$

and the scalar Euclidean distance between two points x and y as:

$$s(\mathbf{x}, \mathbf{y}) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \ldots + (x_{dim} - y_{dim})^2}$$

A transformation between them is defined as:

$$v2s(a) = ||a||$$

with $|| \cdot ||$ the l^2-norm of a vector.

The data structures used by Algorithm 5 are initialised in lines 1, 2, and 3. They include a FIFO queue of active nodes and an array of vector distances. The queue of active nodes is initialised with the subset of nodes defining the distance map origin. Throughout the execution of the algorithm those nodes will be removed from the queue as they are processed (line 5) and new ones will be added if necessary (line 14). The algorithm will finish once the queue is empty. The second data structure is an array of vector distances from each of the nodes to the origin of the distance map used to compute the distance map incrementally. The distances are processed coordinate-wise. This representation is preferred because it captures the evolution of the path between two nodes in contrast with the scalar Euclidean distance that only accounts for the total length of this path. Figure 7.2 illustrates it with a 2D example.

Algorithm 5 Distance map calculation.

INPUT: Let $M = (N, E)$ be a computational mesh, where $N \subset \mathbb{R}^k$ is the set of nodes, $E \subset N^{k+1}$ is the set of elements, and k is the dimension of the problem. Let $S \subset N$ be the origin of the distance map.

OUTPUT: Let $D = \{dist_n \in \mathbb{R} : n \in N\}$ be the scalar Euclidean distances from every node in N to S.

DATA STRUCTURES: Let $V = \{v_n \in \mathbb{R}^k : n \in N\}$ be an array of vector Euclidean distances for all the nodes in N. Let *active_nodes* be a queue of nodes with operators pop() and push() defining the dequeue and enqueue operations respectively.

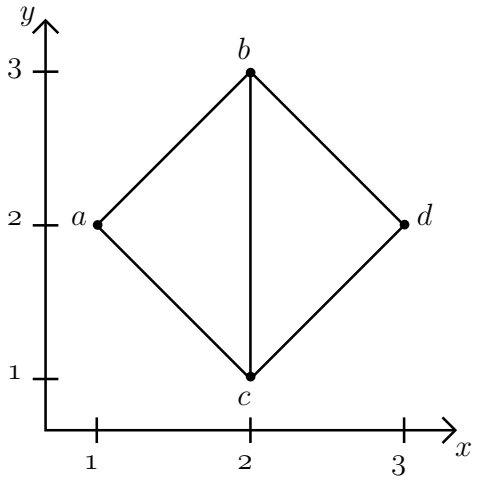
1. $active\_nodes := S$
2. $\forall n \in N - S, V(n) := (\infty, \infty, \ldots, \infty)$
3. $\forall n \in S, V(n) := (0, 0, \ldots, 0)$

4. **while** $active\_nodes \neq \emptyset$
5. target_node:=active_nodes.pop()
6. $\forall e \in E : (e = \{n_{i \in \{1,\ldots,k+1\}}\} \wedge \exists j : target\_node = n_j)$ **do**
7. $\forall n_l \in N : (e = \{n_{l \in \{1,\ldots,k+1\}}\} \wedge n_l \neq target\_node)$ **do**
8. $neighbour\_node := n_l$
9. $target\_euclidean := v2s(V[target\_node])$
10. $neighbour\_euclidean := v2s(V[neighbour\_node])$
11. $target\_to\_neighbour := s(target\_node, neighbour\_node)$

12. **if** $(target\_euclidean + target\_to\_neighbour < neighbour\_euclidean)$
13. $V(neighbour\_node) :=$
$V(target\_node) + v(target\_node, neighbour\_node)$
14. $active\_nodes.push(neighbour\_node)$
15. **end if**
16. **end** $\forall$
16. **end** $\forall$
17. **end while**

18. $\forall n \in N, D(n) := v2s(V[n])$

Figure 7.2 Distance between a and d is 2 whereas length of the shortest path between them is $2\sqrt{2}$.



Let a be the origin of the distance map. In the first iteration of the algorithm, b and c will be added to the list of active nodes and their vector distances to the origin will be set to

$$V[b] = V[a] + v(a, b) = (0, 0) + (1, 2) - (2, 3) = (-1, -1)$$

$$V[c] = V[a] + v(a, c) = (0, 0) + (1, 2) - (2, 1) = (-1, 1)$$

In the second iteration b will be processed and the distance from d to the origin will be set to

$$V[d] = V[b] + v(b, d) = (-1, -1) + (2, 3) - (3, 2) = (-2, 0)$$

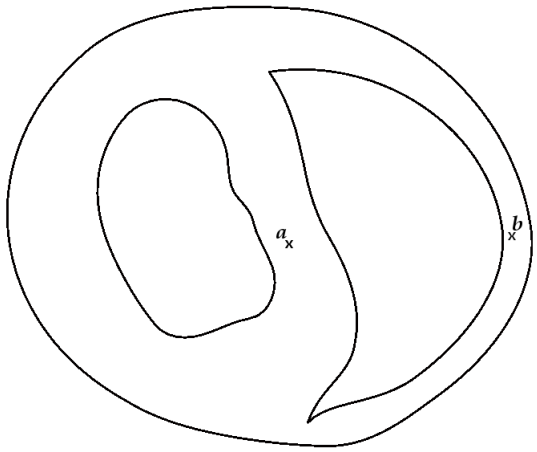
The third and forth iteration of the algorithm will process c and d without modifying V. Finally, the Euclidean distances will be computed

$$D = \{0, \sqrt{2}, \sqrt{2}, 2\}$$

If scalar Euclidean distance had been used throughout the algorithm, it would

have been found that the distance between d and the origin of the distance map is $s(a,b) + s(b,d) = 2\sqrt{2}$. This is an error of more than 40%. This error would propagate to all the nodes that have d in their path to the origin of the distance map and it would likely increase if similar topologies were found. This difference becomes of crucial importance when the algorithm is applied to realistic cardiac geometries, since they are non-convex by nature. Figure 7.3 shows an example.

Figure 7.3 Cardiac geometries are non-convex domains. Length of the shortest path between a and b is much longer than $\overline{ab}$.



Assume that the distance between a and b needs to be computed. Due to the presence of the right ventricle chamber, the length of the line segment $\overline{ab}$ is much shorter than the length of the shortest path between a and b. This is not a problem when distance map algorithms are applied to images (regular grids) since they do not include this kind of discontinuities. In fact, most of the literature reviewed assumes convex domains in the formulation of the algorithms. Little attention is paid to the non-convex case, only a work by Piper and Granum [Piper and Granum, 1987] was found where the non-convex case is studied. One of the authors' conclusions is that propagation algorithms perform better than any other of the options available.

The main loop in Algorithm 5 (lines 4 to 17) visits the nodes in the queue one at a time. The node at the beginning of the queue is considered the target node in the current iteration (line 5). For all its neighbours (defined as the nodes contained in any of the elements that the target node is contained in, lines 6 and 7) the algorithm will check whether its distance to the origin is larger than the distance to the origin of the target node plus the distance between both nodes (line 12). If that is the case, the algorithm will update the distance of the neighbour node with the vector distance of the target node plus the distance between them (line 13) and will add the neighbour node to the list of active nodes (line 14). In the last step of the algorithm (line 18), all the vector distances are converted into scalar Euclidean distances and returned as the output of the algorithm.

As described in [Fabbri et al., 2008], the evolution of the algorithm can be explained by means of the grass-fire analogy. Imagine that the computational mesh represents a network of grass fields. Suppose that fire is started in the set of nodes considered as the origin of the distance map. Fire propagates between nodes through the edges defined by the mesh elements. As the grass gets burned, the fire front gets progressively distant from its initial position, until it extinguishes. It can be said that in time t, the fire will be at some distance d from the origin of the fire. Distances will be computed from the closest node to the furthest. The processing is performed only around the narrow band of nodes (fire front) where a change in the currently stored distance can occur.

Algorithm 5 was successfully applied to compute the distances between the nodes in the ventricular mesh presented in Section 7.1 and the surfaces defining epicardium, left ventricle endocardium, and right ventricle endocardium. In the next subsection, we will explain how based on these data we define fibre orientation for our ventricular model based on the work of [Streeter et al., 1969].

7.2.2 Fibre orientation generation.

Fibre orientation is known to play a crucial role in cardiac electrical activation. When new cardiac geometrical models are developed, DT-MRI images of the heart can be used to estimate fibre orientation. Unfortunately, due to the technical complexity of combining multiple imaging techniques, these are not always available. To overcome this, we have implemented a variation of the mathematical model for ventricular fibre orientation originally presented by [Streeter et al., 1969]. This model, based on the observations over a set of 18 dog hearts, describes how fibres are arranged in the canine left ventricular wall. The ventricular wall is believed to have a well-ordered distribution of fibre angles varying from about 60° at the endocardium to about -60° on the epicardium (although these figures are disputed across the literature). The fibre angle changes smoothly across the wall in both systole and diastole, despite the increase in wall thickness. Generally, the change in angle with respect to the normalised wall thickness is greater near the endocardial and epicardial surfaces. Streeter does not give any analytical formulation for this transition. [Potse et al., 2006] fits Streeter's data with a cubic polynomial. We will use that definition of the fibre angle in our work:

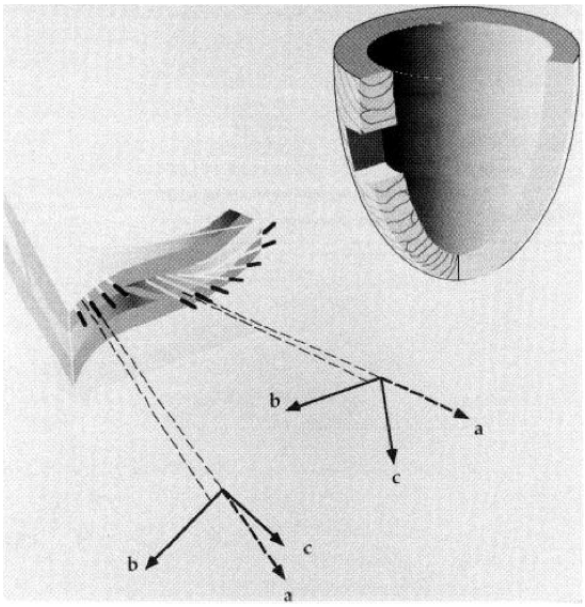
$$\alpha = R(1 - 2p)^3 \tag{7.1}$$

where R is the maximum rotation of the fibres in absolute value and $p \in [0, 1]$ the relative position of the myocyte in the cardiac wall ($p = 0$ in the endocardial surface and $p = 1$ in the epicardial surface). More comprehensive mathematical models exist, see e.g. [Nielsen et al., 1991], which provide accurate representation of the fibre structure throughout the left and right ventricles as well as the septum.

We saw in Chapter 2 that tissue structure can be defined in terms of three orthogonal vectors (fibre direction, fibre sheet direction, and normal to the sheet).

Furthermore, different conductivities are associated with each of these directions. If we combine the previous two concepts, we can define an orthonormal vector basis $(\mathbf{a_f}, \mathbf{a_l}, \mathbf{a_n})$ at each cardiac myocyte describing the direction of the fibre and laminae the myocyte is contained in, and the normal to the laminae. Figure 7.4 shows it graphically.

Figure 7.4 Laminar organisation of myocardium. Vectors a, b, and c represent fibre direction, laminae direction, and normal to the laminae respectively. Image originally from [Legrice et al., 1997].



Algorithm 6 was designed and implemented to generate orthotropic fibre orientation based on the original Streeter description and the refinements described above. Firstly, the algorithm computes the distances between all the nodes in N and the surfaces defining the left ventricle cavity, the right ventricle cavity and the epicardial surface (lines 1 to 3). Based on these distances, the relative transmural position of the nodes p_n is estimated (lines 5 to 6). In order to ensure smooth variation of p_n in areas that sit in the border between different heart regions (i.e. left, right ventricle or septum), p_n is averaged with the value of its neighbours (line 9).

The discretisation of the mono/bidomain equations that we use assumes that conductivity tensors are constant at each element. Hence, we define fibre orientation

Algorithm 6 Generation of fibre orientation data.

INPUT: Let $M = (N, E)$ be a cardiac mesh, where $N \subset \mathbb{R}^3$ is the set of nodes and $E \subset N^4$ is the set of elements. Let $LV, RV, EPI \subset N$ be the nodes located in the left, right ventricle, and epicardial surface of the model respectively. The algorithm assumes that $\hat{\imath} = (1\ 0\ 0)$ is the apex-base direction.

OUTPUT: Let $\{(\mathbf{a_f}\ \mathbf{a_l}\ \mathbf{a_n})_{e \in E}\}$ be the set of orthonormal vectors defining the direction of: fibre, laminae sheet, and laminae sheet-normal for each element in the mesh.

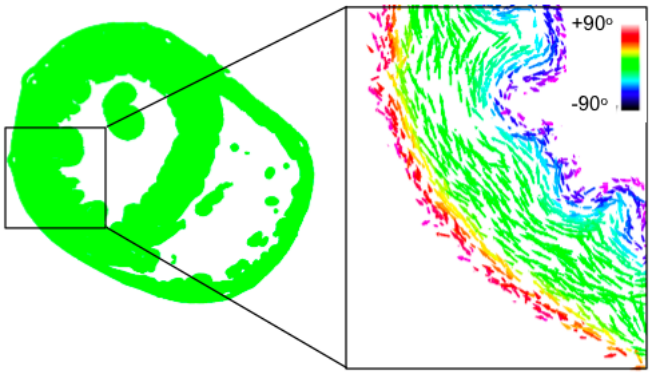
1. $\{d^n_{epi} : n \in N\} := Algorithm\_5(M, EPI)$
2. $\{d^n_{lv} : n \in N\} := Algorithm\_5(M, LV)$
3. $\{d^n_{rv} : n \in N\} := Algorithm\_5(M, RV)$
4. $\forall n \in N$ **do**
5. $\quad d_{endo} := min(d^n_{lv}, d^n_{rv})$
6. $\quad p_n := \frac{d_{endo}}{d_{endo}+d^n_{epi}}$
7. **end** $\forall$
8. $\forall n \in N$ **do**
9. $\quad p'_n := \frac{p_n + \sum_{i \in neig(n)} p_i}{1+|neig(n)|}$, $neig(n)$ is the set of neighbour nodes of n.
10. **end** $\forall$
11. $\forall e \in E$ **do**
12. $\quad \mathbf{u} := \nabla p'$, the gradient of p' at the centroid of the element.
13. $\quad \mathbf{u} := \frac{\mathbf{u}}{||\mathbf{u}||}$
14. $\quad \mathbf{v} := \mathbf{u} \times \hat{\imath}$
15. $\quad \mathbf{v} := \frac{\mathbf{v}}{||\mathbf{v}||}$
16. $\quad \mathbf{w} := \mathbf{u} \times \mathbf{v}$
17. $\quad rv := 0,\ lv := 0$
18. $\quad \forall n \in e$ **do**
19. $\quad\quad$ **if** $(n \in LV)$ **then**
20. $\quad\quad\quad lv := lv + 1$
21. $\quad\quad$ **else**
22. $\quad\quad\quad rv := rv + 1$
23. $\quad\quad$ **end if**
24. $\quad$ **end** $\forall$
25. $\quad$ **if** $(rv > lv)$ **then**
26. $\quad\quad R := \pi/4$
27. $\quad$ **else**
28. $\quad\quad R := \pi/3$
29. $\quad$ **end if**
30. $\quad p'_e := (\sum_{n \in e} p'_n)/4$
31. $\quad \alpha := R(1 - 2p'_e)^3$
32. $\quad (\mathbf{u}\ \mathbf{v_r}\ \mathbf{w_r}) := (\mathbf{u}\ \mathbf{v}\ \mathbf{w}) \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{pmatrix}$
33. $\quad (\mathbf{a_f}\ \mathbf{a_l}\ \mathbf{a_n})_e := (\mathbf{v_r}\ \mathbf{u}\ \mathbf{w_r})$
34. **end** $\forall$

at the centroid of each element. The gradient of the transmural position will give the direction of the laminae $\mathbf{u}$ (lines 12 and 13). The vector product of $\mathbf{u}$ and the apex-base direction will define the direction of the fibre before it is rotated according to (7.1) (line 14). Note that these two vectors are not necessarily orthogonal and therefore their product needs to be normalised (line 15). In a similar way, the non-rotated laminae-normal direction will be defined as the vector product of $\mathbf{u}$ and $\mathbf{v}$ (line 16).

The range of angle variation across the laminae is considered to be different in the left and the right ventricles. An element is considered to be located in the same region as the majority of its nodes (lines 18 to 24). We will consider fibre angles $[-45°, 45°]$ in the right and $[-60°, 60°]$ in the left ventricle (lines 25 to 29).

Finally, the relative transmural position of the element centroid is computed as the average of the relative transmural positions of all its nodes (line 30) and $(\mathbf{u}\ \mathbf{v_r}\ \mathbf{w_r})$ is rotated according to (7.1) (lines 31 and 32). Figure 7.5 shows the output of the previous algorithm when applied to a realistic cardiac geometry.

Figure 7.5 Detail of the fibre orientation data generated for a high-detail rabbit mesh. Vectors represent the direction of the cardiac myofibre. The colour scale represents out-of-plane component of the vector. Image courtesy of Dr Martin Bishop.



Based on the output of Algorithm 6 and the intracellular and extracellular conductivities ($g_f^{i/e}$, $g_f^{i/e}$, $g_f^{i/e}$), the conductivity tensors in (3.84) and (3.85) can be

formulated as:

$$\sigma_{i/e} = (\mathbf{a_f}\ \mathbf{a_l}\ \mathbf{a_n}) \begin{pmatrix} g_f^{i/e} & 0 & 0 \\ 0 & g_l^{i/e} & 0 \\ 0 & 0 & g_n^{i/e} \end{pmatrix} (\mathbf{a_f}\ \mathbf{a_l}\ \mathbf{a_n})^T$$

The implementation of Algorithm 6 in Chaste is generic enough to work with any tetrahedral mesh describing a ventricular geometry.

In this section we have presented tools for the generation of fibre orientation for the human ventricular model described in Section 7.1. These characteristics need to be taken into account in order to reproduce accurately tissue response to shock with the bidomain model. The following section presents the simulation study of shock-induced arrhythmogenesis in the human heart.

7.3 Simulation study of shock-induced arrhythmogenesis in human cardiac geometries

As we saw in Chapter 2, an electrical shock can induce ventricular fibrillation when applied within the period of time known as vulnerable window (VW) following the beginning of a cardiac cycle. Furthermore, arrhythmias can only be induced by shock strengths between a lower and a upper limit of vulnerability (LLV and ULV). A shock applied within the VW and with a strength between LLV and ULV is said to be within the vulnerability area (VA). The mechanisms by which defibrillation shocks re-initiate VF are strongly linked to the mechanisms by which a shock induces VF if applied within the VA following pacing. It is therefore of critical importance for the correct treatment of arrhythmias to be able to characterise VA.

All the simulation studies concerning VA characterisation cited in Chapter 2 were performed with rabbit ventricular geometries because: i) the reduced size of

the rabbit heart decreases the computational burden of the simulations, and ii) lack of availability of human cardiac geometries at the appropriate resolution.

In this Thesis, we have successfully overcome both of those limitations. Firstly, we brought Chaste to the level of performance and functionality required to run simulations with state-of-the-art ventricular models using the novel techniques described in Chapters 4–6. Secondly, a geometrical model of the human ventricles was developed and extended to include tissue anisotropy (which is known to play an important role in defibrillation processes) in Section 7.2. Although the model does not account for some tissue properties relevant to the application such as electroporation, microstructure or action potential heterogeneities, we believe that it includes the basic features necessary to be able to perform a preliminary characterisation of the human vulnerability area.

7.3.1 Methods

Computational model

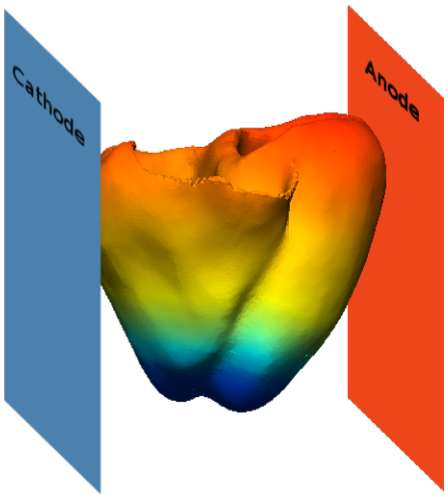
In this study we use the anatomically-based human ventricular model described in Section 7.1. Since information regarding fibre orientation could not be recovered from the original images, we used the generation algorithm presented in Section 7.2. The software pipeline developed by the mesh author allowed for semi-automatic definition of the different mesh surfaces required by Algorithm 6. Transmembrane kinetics are described with the Ten Tusscher 2006 ionic model [ten Tusscher and Panfilov, 2006] (See Section 2.2.1 for a description). The model included representation of the blood in the ventricular cavities and the perfusing bath. Chaste's bidomain solver was used to simulate electrical activity across tissue and bath. The baseline conductivities were $\sigma_b = 8.0$ mS/cm, $\sigma_i = diag(1.7, 0.19, 0.19)$ mS/cm, and $\sigma_e = diag(6.2, 2.4, 2.4)$ mS/cm, where $diag(x, y, z)$ is a 3×3 diagonal matrix with values x, y, z along the diagonal. These set of conductivities were scaled to

reproduce the conduction velocity of 65 cm/s reported in [ten Tusscher and Panfilov, 2006] in the apex-base direction. Other parameters were $\chi = 1400$ cm^{-1} and $\mathcal{C} = 1.0$ μF/cm^2. PDE and ODE timestep were 0.1 and 0.01 ms respectively. The linear solver used was CG with the parallel LDU preconditioner presented in Chapters 5–6. The simulations were run in HECToR phase2a (see Section 6.4 for a description of the hardware).

Protocol for determining vulnerability grids

The protocol used for determining the vulnerability grid was taken from similar studies in the rabbit heart [Rodríguez et al., 2005; Rodríguez and Trayanova, 2003]. The human ventricular model was first paced at the apex by means of an intracellular volumetric stimulus applied to all the cells below the $z = 1.4$ cm plane. Squared monophasic shocks of variable strength and 8ms duration were delivered at varying coupling intervals (CI) via two external plate electrodes located at the boundaries of the bath on the sagittal plane (Figure 7.6). The electrode closer to the right ventricle (RV) was the cathode and the electrode closer to the left ventricle (LV) was the anode.

Figure 7.6 Anterior view of the model with representation of the electrode plates placed at the boundaries of the bath.



The criteria for considering a reentry as sustained was also taken from the literature [Rodríguez et al., 2005]. In brief, the arrhythmia was considered non-sustained if it consisted of one or two extra beats following the shock. However, if a third reentry was observed the arrhythmia was considered sustained. The vulnerability window was considered to be the range of CI for which sustained arrhythmia was induced.

7.3.2 Results

Vulnerability grid

Table 7.1 tabulates the vulnerability grid of the ventricular defibrillation model developed. The symbols in the table represent: i) EB, an extra beat of non-reentrant nature induced by the shock, ii) NS, non-sustained arrhythmia according to the criteria described in the Methods subsection, and iii) F8, figure-of-eight reentry.

Table 7.1 Simulation outcome for a range of shock strengths and coupling intervals.

	290 ms	305 ms	320 ms	335 ms	350 ms	365 ms	390 ms
26 A/cm^2	EB	NS	NS	NS	NS	NS	EB
31 A/cm^2	EB	NS	NS	NS	NS	NS	EB
36 A/cm^2	EB	F8	F8	F8	F8	F8	EB
41 A/cm^2	EB	F8	F8	F8	F8	F8	EB

The vulnerability window of the model spans a range of CI between 297.5±7.5 ms and 372.5 ± 7.5 ms. The lower limit of vulnerability is 33.5 ± 2.5 A/cm^2. The upper limit of vulnerability could not be determined with the range of shocks strengths considered in this study.

Mechanisms leading to reentrant circuits

Figures 7.7–7.9 present the three mechanisms of reentry induction observed. All of them display figure-of-eight morphology, however important differences in location

and number of pathways are observed. All the sustained reentries observed lie within one of these three categories. The different panels in each figure represent the distribution of transmembrane potentials at pre-shock, shock-end, 20 ms post-shock, 80 ms post-shock, and other times relevant for the understanding of the type of reentry induced. In some cases, multiple snapshots of a particular time are given (arranged in rows). Anterior views are presented in all the cases and transmural cuts are depicted at shock-end and 20 ms post-shock. Finally, either apical views or views of the RV septum are showed to highlight the morphology of the reentry.

In agreement with previous studies [Rodríguez and Trayanova, 2003] in rabbit, post-shock transmembrane potential distribution displays the well-known virtual electrode polarisation (VEP) pattern. On the epicardium, two areas of opposite polarisation can be observed: RV epicardium is depolarised while LV epicardium is hyperpolarised. This corresponds to the opposite polarity of the closest electrode (Figure 7.6). Sharp gradients appear at the interface between both areas with the location determined by the shape of the ventricles. The magnitude of the gradient is related to the stimulus strength. We will later see how these gradients are fundamental for the initiation of reentrant waves.

In all our simulations, transmural response to the shock is much more complex than what was observed in the epicardium. At shock-end, the LV wall and the septum exhibit large areas of repolarised tissue with islands of excitation. The location and size of these areas is a function of CI and shock strength, respectively. The isolated areas of depolarised tissue are a result of the use of the bidomain model with anisotropic conductivity ratios and rotating fibres in a constant electrical field [Trayanova et al., 2006]. In contrast, the RV wall is mostly depolarised at shock-end.

Figure 7.7 shows the outcome of a shock of strength 36 A/cm^2 applied at CI of 305 ms. In this case, a F8 reentry with two rotors at the apex is induced. Following the shock, the excitable gap in the RV wall quickly closes due to break excitations

along the RV wall, reaching complete refractoriness by t=20 ms. At the same time, the sharp gradients at the apex initiate a depolarisation wavefront that finds its way through the excitable gap in the LV wall. By t=80 ms the wavefront has already developed the usual reentrant morphology. In the t=190 ms panel we can observe the complex interaction of the wavefront with the apical geometry. Finally, at t=308 ms the wavefront has initiated its second reentrant circuit.

Figure 7.7 Figure-of-eight reentry with rotors at the apex. Shock strength 36 A/cm^2, CI=305 ms.

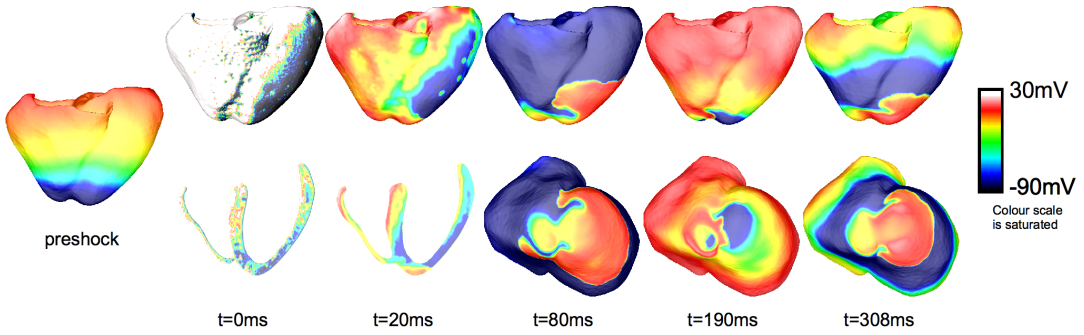


Figure 7.8 shows the outcome of a shock of strength 41 A/cm^2 applied at CI of 335 ms. In this case, three rotors are induced in the anterior LV wall, posterior LV wall and the septum, respectively. Similar to the previous example, the excitable gap in the RV quickly closes following shock-end. At t=20 ms, a depolarisation front initiated at the apex finds its way up the LV wall. In the following 60 ms, the original wavefront becomes three reentrant waves anchored at the anterior LV wall, the posterior LV wall and the septum. Panels t=142 and t=194 ms show how the interaction of the septal reentry and the other two is complex and could be easily mistaken for ectopical activation. At t=194 ms the apical reentry terminates, however at t=324 ms when the two main reentries start their second cycle the septal reentry is restarted again.

Figure 7.8 Figure-of-eight reentry with rotors in the anterior LV wall, posterior LV wall and septum. Shock strength 41 A/cm^2, CI=335 ms.

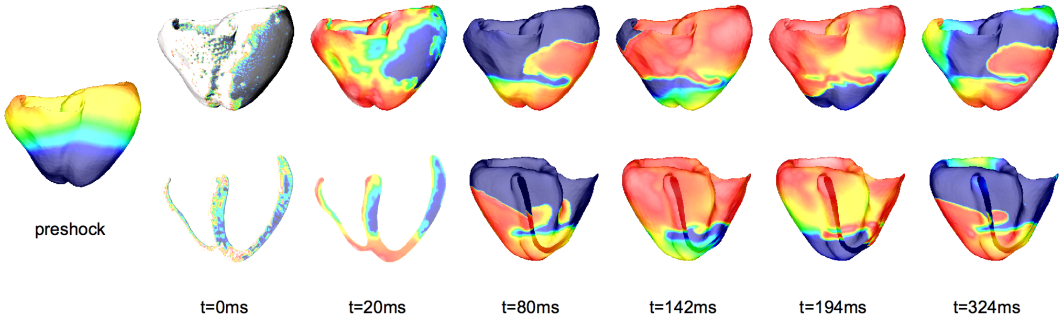
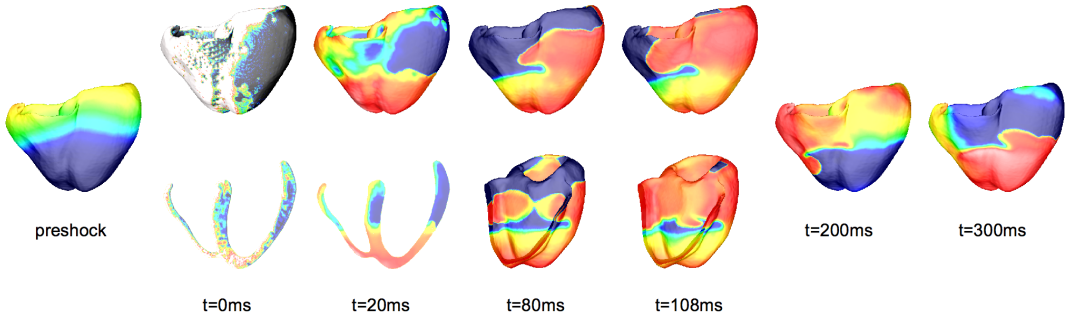


Figure 7.9 shows the outcome of a shock of strength 36 A/cm^2 applied at CI of 350 ms. In this case, two rotors are induced on the anterior and posterior sides of the ventricles, respectively, with a common pathway at the apex. The morphology of this reentry is similar to the previous one, but at t=80 ms the original wavefront has found four propagation fronts: one in the anterior side of the ventricules, one in the posterior side of the ventricles, and two in the septum. The two fronts in the septum will soon collide and become one (t=108 ms) that will eventually terminate at around t=200 ms. This time, the septal reentry is not reinitiated by the principal wavefronts when they start a new cycle (t=300 ms).

7.4 Conclusions

In this chapter we presented novel modeling and simulation techniques for the study of shock-induced arrhythmogenesis in the human heart. Based on a patient specific geometrical model of the human ventricules, we developed a bidomain model of human shock-induced arrhythmogenesis including automatic fibre orientation definition and representation of the blood in the ventricular cavities and the perfusing

Figure 7.9 Figure-of-eight reentry with rotors in the anterior and posterior LV wall. Shock strength 36 A/cm^2, CI=350 ms.



bath. The improvements to Chaste's numerical and parallel efficiency described in Chapters 4–6 allowed performing each of the simulations in 1.5–2.5 hours using 512 cores of HECToR (execution time depends on CI).

The results show that sustained shock-induced arrhythmias occur for CI between 297.5±7.5 ms and 372.5±7.5 ms for an APD90 of 306 ms and an average conduction velocity of 65 cm/s. Shock-end VEP on the epicardium displays two main areas of opposite polarisation with boundaries between the two determined by the shape of the ventricles and the apex, as described in the patient-specific anatomical mesh. Therefore, the accurate representation of ventricular anatomy is critical and was enabled by the use of sophisticated techniques for image acquisition and processing as well as mesh generation. Shock-end VEP in the depth of the ventricular wall shows complex patterns with a large excitable gap located both in the septum and the LV. Shock-induced arrhythmias in the human model consist of figure-of-eight reentry with one rotor in the anterior and one rotor in the posterior of the ventricles. In some cases, a reentry is also established in the septum due to the large shock-induced excitable gap.

In previous studies in rabbit [Maharaj et al., 2008; Rodríguez et al., 2005;

Rodríguez and Trayanova, 2003] the vulnerability windows included CI in the range [70,130] ms, significantly shorter than in this study in human. This is due to differences in APD (96 ms in rabbit vs 250 ms in human). The study also shows that differences in size of the ventricular walls affect VEP and postshock behaviour. Previous studies in rabbit showed how the post shock excitable gap in the RV and septum disappears due to break excitation in areas with large gradients. In this study, we saw how this is still the case in the RV free wall but not in the septum, since the increased wall size leads to less sharp gradients. In our study, shock-induced arrhythmias in the human ventricles consist of figure-of-eight reentry, which is also the most commont type of reentry in studies of shock-induced arrhythmogenesis in rabbit.

The human model presented incorporates realistic ventricular anatomy, membrane kinetics and fibre orientation, key features for the study of mechanisms of shock-induced arrhythmogenesis. The main limitations of the model are:

- The role of cell membrane electroporation has not been taken into account. Electroporation is known to affect cardiac myocyte response to shocks by limiting the maximum transmembrane potential (e.g. see [DeBruin and Krassowska, 1999]). This may influence the formation of the transmembrane potential gradients necessary for break excitation to start.

- Structural heterogeneities such as cleavage planes have not been incorporated into the tissue model. Recent publications [Hooks et al., 2002; Trew et al., 2009] have shown that they provide a substrate for bulk activation of the ventricles during defibrillation and need to be considered in order to understand successful defibrillation.

- The fibre orientation model used is based on measurements in canine left ventricle free wall [Streeter et al., 1969]. There exist more comprehensive models of cardiac microstructure (e.g. see [Nielsen et al., 1991]), which consider

different fibre distributions across left ventricle, right ventricle, and septum as well as localised singularities. Changes in fibre orientation contribute to VEP formation and therefore this added fibre orientation complexity is likely to have an impact.

All these improvements can be the focus of further studies. The simulations presented in this study determined the LLV to be $33.5 \pm 2.5 \mathrm{A/cm}^2$, however the range of shock strengths studied could not determine the ULV which can also be the target of future studies.

This study has also validated the developments undertaken in Chapters 4–6. The set of 28 simulations run took an estimated CPU total time of 56h on 512 cores of HECToR phase2a. Using 2048 cores could reduce execution time by a factor of over two times. Taking into account that, depending on the load of the system, multiple experiments can run concurrently, we conclude that the simulation study presented in this chapter could be run in less than 20 hours of wall clock time

Chapter 8

Conclusions and future work

In this Thesis, we present novel numerical and computational techniques for the efficient solution of the bidomain equations of cardiac electrophysiology in modern HPC hardware. This includes: parallelisation via domain decomposition, communication reduction in the underlying FEM solver, development of less communication-intensive linear solvers and development of efficient sequential and parallel bidomain preconditioners. All of these advances have been implemented in the framework of the open source simulation package Chaste, which makes them readily available to the scientific community.

These advances have brought Chaste to the level of performance and functionality required to run bidomain simulations with large three-dimensional cardiac geometries made of tens of millions of nodes and including accurate representation of fibre orientation and membrane kinetics. The improvements developed have enabled the in-silico study of shock-induced arrhythmogenesis for the first time in the human heart.

The rest of this chapter is structured as follows: Section 8.1 summarises the results presented in Chapters 4–6, Section 8.2 describes future lines of work and finally we end the chapter, and the Thesis, in Section 8.3 with some concluding remarks.

8.1 Conclusions

The results presented in this Thesis can be grouped into three main areas: i) Chaste's parallel scalability improvements, ii) sequential and parallel bidomain preconditioning, and iii) in-silico study of shock-induced arrhythmogenesis in the human heart. This section summarises each of these areas.

8.1.1 Chaste's parallel scalability improvements

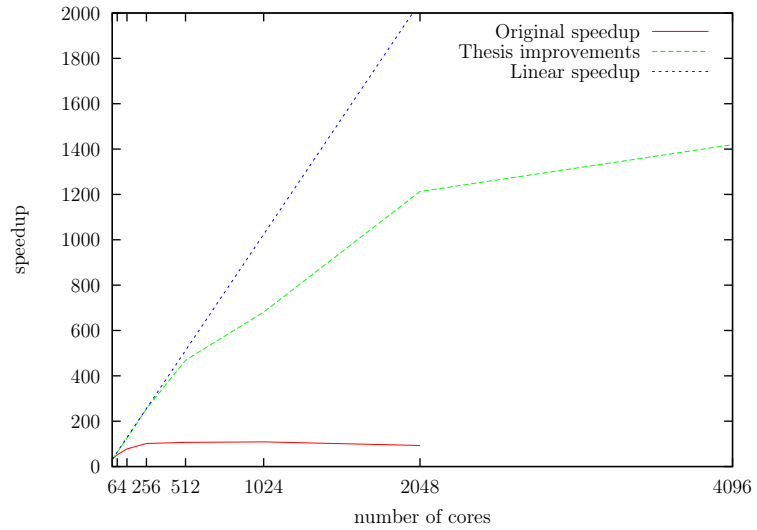
Chapter 4 presents effective parallelisation strategies for all of the stages involved in a bidomain simulation with Chaste. Prior to the work presented in this Thesis, Chaste could not run bidomain simulations with geometrical models at the level of detail required by our application of interest. This was mainly due to the lack of an appropriate approach to domain decomposition of the data structures representing the mesh in memory. Domain decomposition was first implemented and graph partitioning-based techniques were used to improve scalability at lower core counts (72% efficiency was achieved with 32 cores). This allowed us to run Chaste simulations with geometrical models at para-cellular resolution [Bernabeu et al., 2008, 2009]. However, the simulation study presented in Chapter 7 requires: i) simulations of a duration that we had never considered before (i.e. in the order of seconds), and ii) the ability to run a large number of simulations in order to characterise the cardiac response to electric shocks. Performing this study required, therefore, the use of large scale HPC resources such as HECToR. Chaste was successfully ported to HECToR and initial profiling highlighted scalability issues that could only be identified at this large scale (e.g. mesh read, RHS assembly, and linear system solution stages). Firstly, the mesh read and partitioning time was improved with the implementation of direct file access, the use of parallel domain decomposition algorithms, and the re-design of the file format used for representing meshes. For this stage, 62% efficiency was achieved with 256 processors and also yielding an over

50-fold reduction of load time on 4096 processors when compared with the original implementation. Secondly, matrix assembly scalability was improved by introducing a small amount of computation redundancy across processors in order to ensure that the portion of the matrix assigned to each processor was generated locally, avoiding data migration. Benchmarks show linear speedup with up to 512 processors and 72% scalability for 4096 processors. Following the RHS assembly optimisation described in [Pathmanathan et al., 2010], the RHS assembly stage was further sped up by removing unnecessary communications and synchronisations, yielding a 350-fold improvement over the original version for 4096 processors. Finally, the hybrid CG-Chebyshev linear solver proposed in this work is 52% faster than standard linear solvers such as CG for 4096 processors. This improvement is crucial since — after all other stages were optimised — linear system solution takes up to 80% of the total execution time.

All of these optimisations combined yield a speedup of over 1400 and optimal or near-optimal scalability for up to 512 processors (Figure 8.1). The improvements presented in this Thesis yield a combined speedup improvement of over 140, meaning that Chaste's bidomain solver is now two orders of magnitude faster than it was before the improvements presented in this Thesis. This has enabled the simulation study presented in Chapter 7. In absolute terms, Chaste's bidomain solver is 210 times slower than real-time for a simulation with 7.4M degrees of freedom and 4096 processors. This real-time ratio is faster than any of the works surveyed in Chapter 4. Finally, Chaste's monodomain solver is 45 times and 2.8 times slower than real-time for simulations with 3.7M and 64K degrees of freedom, respectively. This leaves the milestone of being able to run faster than real-time simulations of cardiac electrophysiology within reach, if only by relying on hardware advances in the near future.

In this work, we have presented bidomain simulations with up to 4096 cores in

Figure 8.1 Chaste's scalability for a typical bidomain simulation before and after this Thesis.



HECToR. With the consolidation of petascale computing, current trends in HPC are moving towards machines with two orders of magnitude more available cores (e.g. Japan's K computer with 705024; the fastest machine in the world as of November 2011). We consider the work in this Thesis a step forward in the direction of being able to fully utilise this new generation architectures. Many open problems lie ahead, though. In Chapter 4, we saw how Chaste's I/O operations become a bottleneck at large scale. Further performance improvements will also be required in some of the underlying numerical kernels (e.g. sparse linear solvers and preconditioners) in order to be able to sustain 10^{15} floating point operations per second or more as available in the latest petascale machines.

8.1.2 Sequential and parallel bidomain preconditioning

Following the scalability improvements summarised in the previous subsection, the linear system solution was the stage taking the largest proportion of total execution time (around 70% with 4096 processors, Chapter 4). Performance of this stage is directly determined by the number of iterations taken by Chaste's iterative linear

solver, which at the same time is dictated by the effectiveness of the preconditioner used. Chapters 5 and 6 present an in-depth study of current bidomain preconditioning techniques and the development of novel mesh-independent methods with particular stress on parallel performance. In Chapter 5, we study the convergence properties of a combination of four preconditioners (incomplete LU factorisation with no fill-in, ILU(0), algebraic multigrid, AMG, block diagonal, BD, and LDU factorisation, LDU) and two formulations of the bidomain equations (parabolic-parabolic, PP, and parabolic-elliptic, PE). For isotropic tissue, the results suggest that when only an intracellular stimulus is used, the low iteration count of CG preconditioned with ILU(0) added to the relatively low computational cost of the technique makes ILU(0) the most efficient preconditioner in the study. The choice of bidomain formulation (PP vs PE) does not affect its convergence rate. The convergence analysis presented illustrates the problem-dependent nature of ILU(0).

Among all the combinations tested, only PE-LDU and PP-LDU display mesh-independent convergence. The iteration count of PE-LDU is lower than its PP counterpart. BD also performs better when applied to PE. However, full AMG shows the opposite behaviour. These results are consistent with [Pennacchio and Simoncini, 2009]. Despite the mesh-independence of PE-LDU, its higher cost per iteration makes it more efficient than PE-ILU only for $h \lesssim 1\mu m$. This value of h is two orders of magnitude smaller than in current realistic geometries and therefore not currently relevant. When considering anisotropic tissue, this result holds when the same anisotropy ratios are used for both intracellular and extracellular conductivity. However, when different ratios are considered in each of the spaces PE-LDU significally outperforms PE-ILU.

In the presence of extracellular shocks, the iteration count of CG preconditioned with ILU(0) increases dramatically. In our experiments, convergence was not reached within the first hundred iterations and therefore simulations were aborted.

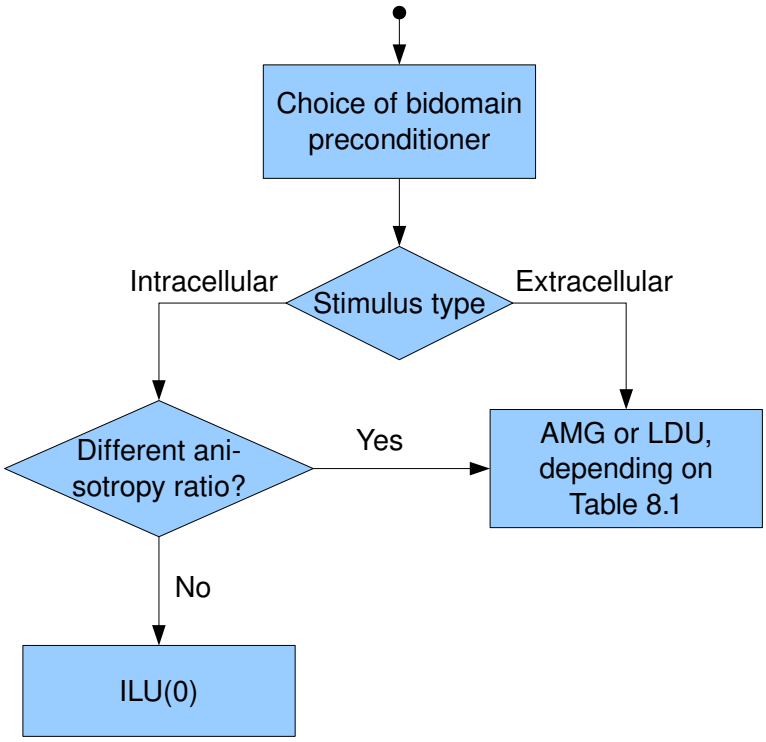
This observation is consistent with [Colli-Franzone and Pavarino, 2004] and [Vigmond et al., 2008]. However, in those publications results are generalised to any stimulus protocol or tissue properties whereas we have characterised under which problem configurations this situation arises (i.e. extracellular shocks or intracellular stimuli with very different anisotropy ratios in the intra- and extracellular domain). In this context the use of more sophisticated preconditioning techniques is mandatory.

The results of the study show how full AMG performs well when applied to small and medium size meshes. However, memory issues arise when it is applied to the high resolution mesh used in our study. The iteration count for BD is around 4 iterations higher than for full AMG. However, its lower cost per iteration makes it quicker in absolute terms. Furthermore, BD works with the high resolution mesh. LDU performs well for the low and high resolution meshes, being quicker than BD in both cases. Figure 8.2 presents a decision diagram for the choice of the most efficient preconditioner.

In Chapter 6, we present and evaluate parallel implementations of BD and LDU. In a sequential simulation, both show promising results when compared to generic preconditioning techniques such as ILU(0) or AMG. However, their parallelisation is challenging because they potentially require data migration for the assembly of intermediate data structures, which can compromise scalability. The implementation proposed uses a data partitioning scheme that ensures no data migration when assembling these intermediate data structures. This guarantees that parallel scalability is only restricted by the underlying inverse approximation algorithms and therefore future scalability improvements to them will automatically improve BD and LDU without the need of re-coding.

In Chapter 5, we considered one AMG V-cycle for the approximation of the action of the inverse of certain matrix subblocks — A_1^{-1} and A_2^{-1} in (5.2) and (5.3)

Figure 8.2 Decision diagram for the choice of the most efficient bidomain preconditioner.



— with HYPRE's BoomerAMG algorithm default configuration. In Chapter 6, we investigate other options: a reduction of 38% of total linear system solution time is achieved by using ILU(0) for the A_1^{-1} approximation and one AMG V-cycle with HMIS coarsening for A_2^{-1}.

The original mesh edge length (h) dependence analysis was extended with a combined edge length and solution timestep (Δt) dependence analysis. Results show a good correlation between large values of $\Delta t/h^2$ and the situations where BD and LDU outperform AMG. This is due to the fact that the condition number of the preconditioned system is a function of $\Delta t/h^2$ when mesh-dependent preconditioners are considered [Wathen, 1987]. Therefore, the number of iterations required by AMG will grow unbounded as the condition number increases. In contrast, the LDU iteration count either: i) converges to an asymptotic bound when one AMG V-cycle is used for the approximation of both sub-blocks, or ii) grows at a smaller rate when the efficacy of the first block inverse approximation is relaxed (see Table 6.4). Table 8.1 gives the value of the coefficient for all of the combinations of Δt and h considered and highlights in dark grey cases where BD and LDU outperform AMG. In our experiments, this happens for values of $\Delta t/h^2$ greater than 2.5. For values below 2.5, the system is not so ill-conditioned and all the preconditioners considered keep the iteration count low. In this scenario, the slightly lower computational cost per iteration of AMG makes it faster than BD and LDU.

Table 8.1 Coefficient $\Delta t/h^2$ for all the combinations of Δt (ms) and h (mm) considered. In dark grey, cases where BD and LDU outperform AMG for any number of processors. In light grey, configurations where this only happens for low core counts

Δt \ h	1	0.5	0.25	0.125	0.0625
0.1	0.1	0.4	1.6	6.4	25.6
0.05	0.05	0.2	0.8	3.2	12.8
0.01	0.01	0.04	0.16	0.64	2.56

Finally, strong scaling of the three methods (Figure 6.1) is good although not

excellent (under 40% with 1024 processors). The reasons are: i) the increase in iteration count with number of processors described in Chapter 6, and ii) the tightly-coupled nature of the approximation algorithms. Furthermore, strong scaling is slightly better for AMG than for BD and LDU (around 55% with the same configuration). Nevertheless, total linear system solution time with BD and LDU is still over 3 times faster than with AMG for the range of core counts considered in the strong scaling study.

8.1.3 Shock-induced arrhythmogenesis in the human heart

The scalability and performance improvements presented in Chapters 4–6 enabled the simulation of shock-induced arrhythmogenesis in realistic human ventricular geometries with Chaste. In Chapter 7, a model of human defibrillation was developed and used to perform the first study of shock-induced arrhythmogenesis in the human heart. The results show that sustained shock-induced arrhythmias occur for coupling intervals between 297.5 ± 7.5 ms and 372.5 ± 7.5 ms for an action potential duration of 306ms and an average conduction velocity of 65cm/s. Shock-end virtual electrode polarisation on the epicardium displays two main areas of opposite polarisation with boundaries between the two determined by the shape of the ventricles and the apex. Shock-end VEP in the depth of the ventricular wall shows complex patterns with a large excitable gap located both in the septum and the LV. Shock-induced arrhythmias consist of figure of eight re-entry with one rotor in the anterior and one rotor in the posterior of the ventricles. In some cases, a re-entry is also established in the septum due to the large shock-induced excitable gap.

In previous studies in rabbit [Maharaj et al., 2008; Rodríguez et al., 2005; Rodríguez and Trayanova, 2003] the vulnerability windows included CI in the range [70,130], significantly shorter than in this study in human. This is due to differences in APD (96ms in rabbit vs 250ms in human). The study also shows that differ-

ences in size of the ventricular walls affect VEP and postshock behaviour. Previous studies in rabbit showed how the post shock excitable gap in the RV and septum disappears due to break excitation in areas with large gradients. In this study, we saw how this is still the case in the RV free wall but not in the septum, since the increased wall size leads to less sharp gradients. The simulation study also shows that shock-induced arrhythmias in the human ventricles consist of figure of eight re-entry, which is also the most common type of re-entry in studies of shock-induced arrhythmogenesis in rabbit.

8.2 Future work

This section presents open problems related to the advances presented in this Thesis as well as cases where the techniques proposed could be applied to other Systems Biology applications.

8.2.1 Load balancing of bidomain problems with a bath

In Chapter 4, we studied the problem of load balancing bidomain simulations without the representation of the external media surrounding the heart (i.e. without a bath). The block structure of linear system (3.104) reveals that the approach proposed is sub-optimal for bidomain simulations with a bath. This is due to the different amount of computation associated with degrees of freedom located in tissue versus those located in the bath. More precisely, when assembling the right-hand side in (3.104), I_{ion} only has to be computed at the tissue nodes. Therefore, if a processor is assigned a partition entirely made of bath nodes, it will remain idle waiting for other processors owning the tissue nodes to finish before moving into the next simulation stage. Solving the linear system (3.104) is also potentially imbalanced since rows corresponding to values of V in the bath have fewer non-zero entries.

Therefore, linear solvers that take into account the sparsity pattern of the system matrix will perform fewer operations in subdomains mainly made of bath nodes.

The graph-partitioning algorithm used for domain decomposition in Chapter 4 can be extended to assign different weights to different parts of the domain based on computation requirements. Therefore, a characterisation of the load associated with tissue and bath nodes needs to be performed and integrated into the METIS-based algorithm by means of assigning different weights to the nodes and/or edges of the graph representing the mesh.

8.2.2 Error reduction in the hybrid CG-Chebyshev linear solver

Figure 4.8 shows how the error associated with the use of a fixed number of iterations in the hybrid CG-Chebyshev linear solver is larger than the linear solver tolerance during a small portion of the simulation. In the same figure, it can be appreciated that increasing the frequency of tolerance-based solves (i.e. the 1:s ratio) reduces the magnitude of the error introduced. However, this has a negative impact on scalability since the number of l^2-norm operations performed increases.

Superimposing Figures 4.7 and 4.8 highlights the fact that the error comes from sudden increases in the number of iterations associated with events such as the initiation of depolarisation and repolarisation wavefronts, with the former inducing errors of larger magnitude than the latter. A possible improvement to the method would be to perform tolerance-based solves for as long as any stimulus is applied, switching to the hybrid schema once the stimulation is removed. Note that these kind of events are user-defined and therefore known at the beginning of the simulation.

8.2.3 Spatial adaptivity

High resolution computational meshes are required for the accurate simulation of very sharp, but highly localised, propagation wavefronts in cardiac tissue. As discussed throughout this Thesis, the use of high resolution meshes made of large number of nodes results in high computational costs. Several authors [Belhamadia, 2008; Cherry et al., 2003; Deuflhard et al., 2009; Southern et al., 2010; Trangenstein and Kim, 2004] have proposed the use of adaptive mesh methods as a way of reducing computational burden while preserving accuracy. By maintaining the extremely fine resolution only where it is needed (i.e. in the vicinity of the wavefront) and coarsening elsewhere, the number of degrees of freedom could be greatly reduced, resulting in much faster calculations (as well as reduced memory usage and smaller output files). Due to the technical complexity of this task, mixed results have been reported to date. For example, a factor 150 reduction in the number of mesh nodes compared with a non-adaptive method was reported in [Weiser et al., 2008], but this did not translate into an equivalent reduction in computation time. According to the authors, the reasons are: i) the computation of error estimates required by the mesh adaptation algorithm taking a significant portion of the total execution time, ii) the need for recomputing the FEM matrix after each mesh modification, and iii) the cost of the mesh modifications themselves and the complex data structure management overhead.

In [Southern et al., 2010], closed collaborators incorporated a sequential anisotropic mesh adaptivity algorithm into Chaste in order to reduce the mesh resolution away from the depolarisation front. The method was extensively tested in normal sinus rhythm simulations with idealised and realistic geometries. The use of mesh adaptivity results in a reduction in the number of degrees of freedom by an order of magnitude during propagation and 2–3 orders of magnitude in the subsequent plateau phase. As a result, reduction of computation time by a factor of between 5

and 12 is reported with no loss of accuracy. The work is further extended in [Southern et al., 2011], where a parallel version of the adaptivity algorithm is presented and evaluated. Good scaling within the range 1–64 processors is reported for a sinus rhythm simulation. Total execution time is, however, shown to be dominated by the adaptive mesh algorithm and not by the solution of the bidomain equations themselves.

Spatial adaptivity, therefore, constitutes a promising way of reducing computational burden in cardiac electrophysiology simulations. However, further improvements to the adaptive mesh methods currently available are required in order to take full advantage of large-scale computational resources. The methods available also need to be shown competitive in fibrillation scenarios where multiple wavefronts travel across the domain in a disorganised manner. Finally, further research is also required on preserving tissue structure and heterogeneities while the computational mesh is being coarsened or refined.

8.2.4 Tissue heterogeneity and bidomain preconditioning

In Chapter 5, we saw that the choice of anisotropy ratio in the bidomain equations has an impact on the performance of the preconditioning techniques proposed. We provided guidelines on how to choose the most appropriate technique but no formal investigation of the connection between the anisotropy ratio and the matrix spectrum was carried out.

In addition, no evaluation of the impact that other tissue heterogeneities may have on bidomain linear system conditioning has been performed. This includes changes in fibre orientation and heterogenous conductivity values across the domain. In [Pennacchio and Simoncini, 2009], the authors show how the presence of big jumps in conductivity leads to a moderate performance degradation of the AMG and LDU preconditioners in a simple 2D benchmark. From a practical point of view, this

may be become important for applications that require decreasing conductivity in certain parts of the tissue in order to model ischaemic regions or when the body surface ECG is computed with whole-torso models (e.g. see [Zemzemi et al., 2011]) and different conductivities are assigned to different parts of the torso (e.g. bones, organs).

8.2.5 Extended bidomain linear algebra

In a recent publication, [Corrias et al., 2011] present an extension of the bidomain model that allows for multiple cell types to contribute to the tissue electrophysiology. They apply the new formulation to the simulation of cardiac and gastrointestinal electrophysiology. In the heart, the interaction of myocytes and fibroblasts is studied. In the gastrointestinal track, interstitial cells of Cajal and smooth muscle cells are combined.

In the extended bidomain model, the original PDE describing changes in intracellular potential is split into two equations characterising potential within both cell types separately. In order to characterise the proportion of cardiac tissue made of each cell type, the original surface-area-to-volume ratio χ is divided into three constants: $\chi^{(1)}$, $\chi^{(2)}$, and $\chi^{(gap)}$, which correspond to the surface-area-to-volume ratio for cell (1), cell (2), and the space occupied by the gap junctions separating the two intracellular domains, satisfying $\chi = \chi^{(1)} + \chi^{(2)} + \chi^{(gap)}$. Following similar notation

to (3.86)–(3.88), the model can be formulated as:

$$\begin{aligned}
\nabla \cdot \left(\sigma_i^{(1)} \boldsymbol{\nabla} \phi_i^{(1)}\right) &= \chi^{(1)} \left(\mathcal{C}^{(1)} \left(\frac{\partial \phi_i^{(1)}}{\partial t} - \frac{\partial \phi_e}{\partial t} \right) + I_{\text{ion}}^{(1)}(\mathbf{u}^{(1)}, \phi_i^{(1)}, \phi_e) \right) + I_{\text{stim}}^{(1)} + \chi^{(gap)} I_{\text{gap}} \\
\nabla \cdot \left(\sigma_i^{(2)} \boldsymbol{\nabla} \phi_i^{(2)}\right) &= \chi^{(2)} \left(\mathcal{C}^{(2)} \left(\frac{\partial \phi_i^{(2)}}{\partial t} - \frac{\partial \phi_e}{\partial t} \right) + I_{\text{ion}}^{(2)}(\mathbf{u}^{(2)}, \phi_i^{(2)}, \phi_e) \right) + I_{\text{stim}}^{(2)} + \chi^{(gap)} I_{\text{gap}} \\
\nabla \cdot (\sigma_e \boldsymbol{\nabla} \phi_e) &= -\nabla \cdot \left(\sigma_i^{(1)} \boldsymbol{\nabla} \phi_i^{(1)}\right) - \nabla \cdot \left(\sigma_i^{(2)} \boldsymbol{\nabla} \phi_i^{(2)}\right) + I_{\text{stim}}^{\text{total}} \\
\frac{\partial \mathbf{u^{(1)}}}{\partial t} &= \mathbf{f}^{(1)} \left(\mathbf{u}^{(1)}, \phi_i^{(1)}, \phi_e \right) \\
\frac{\partial \mathbf{u^{(2)}}}{\partial t} &= \mathbf{f}^{(2)} \left(\mathbf{u}^{(2)}, \phi_i^{(2)}, \phi_e \right)
\end{aligned}$$

where $I_{\text{stim}}^{\text{total}} = I_{\text{stim}}^{(1)} + I_{\text{stim}}^{(2)} + I_{\text{stim}}^{(e)}$ and the superscripts (1) and (2) correspond to the definition of variables, constants, and operators in each of the two intracellular domains. Appropriate boundary conditions are

$$\begin{aligned}
\mathbf{n} \cdot \left(\sigma_i^{(1)} \boldsymbol{\nabla} \phi_i^{(1)}\right) &= 0 \\
\mathbf{n} \cdot \left(\sigma_i^{(2)} \boldsymbol{\nabla} \phi_i^{(2)}\right) &= 0 \\
\mathbf{n} \cdot (\sigma_e \boldsymbol{\nabla} \phi_e) &= 0
\end{aligned}$$

Following the same semi-implicit time discretisation and finite element space discretisation presented in Chapter 2, one is left with the solution at each timestep of a linear system $A\mathbf{x} = \mathbf{b}$ with $A \in \mathbb{R}^{3N \times 3N}$, $\mathbf{x}, \mathbf{b} \in \mathbb{R}^{3N} : \mathbf{x} = (\Phi_i^{(1)}, \Phi_i^{(2)}, \Phi_e)^T$ and

$$A = \begin{bmatrix}
\frac{\chi^{(1)}\mathcal{C}^{(1)}}{\Delta t} M + K[\sigma_i^{(1)}] & 0 & -\frac{\chi^{(1)}\mathcal{C}^{(1)}}{\Delta t} M \\
0 & \frac{\chi^{(2)}\mathcal{C}^{(2)}}{\Delta t} M + K[\sigma_e^{(2)}] & -\frac{\chi^{(2)}\mathcal{C}^{(2)}}{\Delta t} M \\
K[\sigma_i^{(1)}] & K[\sigma_i^{(2)}] & K[\sigma_e]
\end{bmatrix}$$

where N is the number of nodes in the computational mesh.

In [Corrias et al., 2011], the authors use the GMRES algorithm with point Jacobi preconditioning for the solution of $A\mathbf{x} = \mathbf{b}$. We strongly believe that the precondi-

tioners presented in Chapters 5 and 6 can be adapted to this problem in order to reduce execution time.

8.3 Concluding remarks

Modelling and simulation have become fundamental tools in the field of cardiac electrophysiology. The use of computational models has been particularly fruitful in the area of cardiac defibrillation, with a number of experimental discoveries grounded on theoretical predictions of the bidomain model of cardiac electrophysiology.

Over the past 20 years, the cardiac defibrillation models used for the study of the mechanisms underpinning success and failure of defibrillation shocks have grown in complexity. From simple one-dimensional models to complex anatomically-accurate representations of whole-ventricular geometries, a plethora of works have shed light on the processes governing initiation and termination of life-threatening cardiac arrhythmias by means of electrical shocks. One of the next big milestones is validating which of the theories developed with computational models of small mammalians hold for the human, with the ultimate goal of being able to answer questions of clinical relevance. Up until now, this has not been possible due to limitations in the simulation technology available. In this Thesis, we have developed novel numerical and computational techniques for the solution, in cutting-edge HPC infrastructures, of the bidomain equations over large three-dimensional cardiac geometries including accurate representation of fibre orientation and membrane kinetics. We have shown that the simulation of shock-induced arrhythmogenesis in the human heart is possible.

As computational models of cardiac electrophysiology grow in complexity, the complexity of the simulation platforms supporting them also has to increase. The traditional approach to software development in academia, with individuals developing code of very limited life span, is likely not to meet the requirements. We have

ahead of us a turning point in the way code is developed, engineered, and validated in academic environments. Open sourced collaborative development using state-of-the-art software engineering techniques will enable the next generation of in-silico science. All the solutions presented in this Thesis have been implemented in Chaste, implying that they are immediately available to the scientific community. With over 700 downloads since June 2009[1], Chaste has become a point of reference in Systems Biology research concerning cardiac, cancer, soft-tissue simulation.

In addition, the solutions presented in this Thesis are generic enough to be able to percolate into other areas of computational biology sharing similar underlying mathematical models. Consequently, it is hoped that the techniques presented in this Thesis have provided a valuable contribution to the field of computational biology, and more specifically to cardiac defibrillation research.

[1] `https://chaste.cs.ox.ac.uk/chaste/stats.php`

Appendix A

Agile methods in software engineering

As mentioned in the Introduction, at the beginning of the Chaste project, packages available for cardiac electrophysiological simulation presented limitations. These limitations can be grouped into one of the following topics:

- efficiency,
- robustness,
- extensibility,
- reliability, and
- availability

The selection of software development paradigm plays a crucial role in determining whether a software product will fulfill these, sometimes conflicting, requirements. The working conditions and practices in academia do not facilitate the adoption of good software engineering practices. A typical academic scenario is that of a single person writing code for his or her own research. Often this involves a doctorate student, without any background in software engineering, who is not interested in

maintaining the code or adding functionality after he or she graduates. Once the research which the code was originally supporting is published, it will become unused or, at best, a 'black-box' that is used without modification or maintenance. Difficulties preventing development and maintenance include:

- esoteric coding styles,
- lack of documentation,
- inappropriate testing (if any) or
- incorrect architectural decisions that compromise future development.

If the code is developed by a team of researchers extra problems arise if they work in a loosely coupled manner or without good communication channels:

- integration proves difficult, since local copies of the code base become outdated quickly, and
- if different parts of the codebase are developed individually and those responsible for a given part leave the project, important slowdowns are likely to happen.

It is clear that this software development scenario is far from being ideal and likely to produce low quality software products, tension among developers, and a high degree of frustration.

A.1 Agile programming

This problem was first addressed by the software industry a few decades ago. The solution proposed was bringing methodologies from other fields of engineering into software development. Imposing rigorous processes on software development, with

the aim of making it predictable and more efficient. These methodologies separate the writing of software into design and coding, where the result of design is a plan that is supposed to make coding a predictable activity. In practice, however, plan-driven methodologies are not noted for being successful, require a lot of documentation and are frequently criticised as being too bureaucratic. The control over the development process is so tight that it leaves little to no room for adaptation to the dynamic environment of software development and validation, slowing down the development itself and creating static products [Fowler, 2000].

The separation between design and construction arises naturally in some fields of engineering, since the cost of the design stage is much lower than the production cost. In the best scenario, investing a small quantity of resources in design has a proportionally much higher impact on the construction cost. In the worst case, one cannot afford anything but one construction stage. Unfortunately, this does not apply to software engineering, where design is neither cheap nor static.

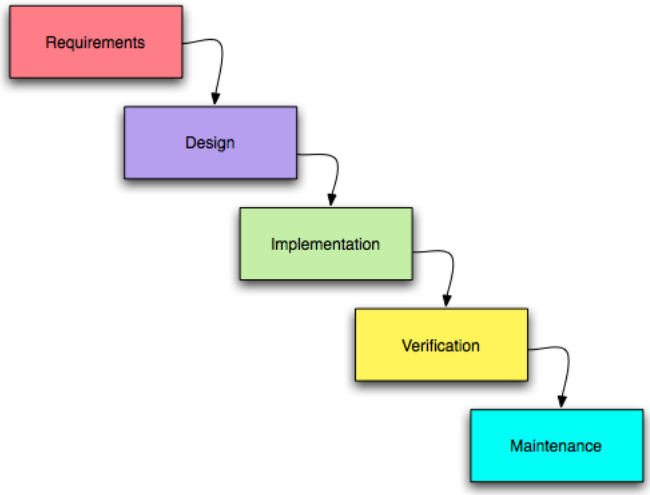
Agile methods can be seen as a reaction to plan-driven methodologies. They pave a third way between no methodology and bureaucratic engineering-inspired methodologies. Rather than attempting to make software development a predictable activity, they try to provide developers with enough tools to be able to adapt to changes in the requirements and circumstances of the project

The key to this adaptive approach is feedback. This is achieved through incremental and iterative software development. Fully tested and working versions of the final product are generated at frequent intervals. The plan-driven approach follows the so-called waterfall workflow (Figure A.1). Since there is no feedback defined between later stages of the process and former ones, substantial resources need to be invested in generating a design that completely satisfies all the requirements. If inaccuracies are discovered in the verification stage, the whole process needs to be restarted at great expense. By contrast, the agile approach calls for implementing

high-priority requirements into a working and usable system quickly. Therefore, important structural disagreements with the requirements will be discovered at earlier stages of the project, before too much cost is incurred. The amount of work in progress is also minimised.

It should be noted that the constantly changing nature of science makes running requirement gathering exercises pretty much impossible. For example, in a plan-driven 2 year project, requirements will be gathered during the first two months and the product will be delivered at the end of month 24. As no extra requirements are allowed into the project after the initial stage, the product will be useful for the kind of science that was state-of-the-art two years ago but not for what it is now. The ultimate goal of publishing will turn into a rather difficult task.

Figure A.1 Waterfall software development workflow. Image originally released under Creative Commons Attribution and ShareAlike license.



In order for agile methods to work, the software coming out of one iteration must not only meet the requirements of the user, but must also have internal qualities so that it may form a solid basis for the next iteration. Software needs to be built so that it can be adapted and extended, perhaps in ways that were not originally thought about at the start of the project. The 'eXtreme Programming' (XP) approach [Beck

and Andres, 2004] adopted by the Chaste project, prescribes a number of engineering practices to ensure that this is the case. The following subsections describe the most common practices in XP.

A.1.1 The customer and user stories

In XP, the customer plays an important role throughout the whole life of the project and not only restricted to the requirements gathering stage. He or she provides user stories describing features of interest. New user stories can come in at any moment and priority metrics can be implemented. User stories replace large requirement documents. The granularity of the user stories is important to minimise the amount of work in progress.

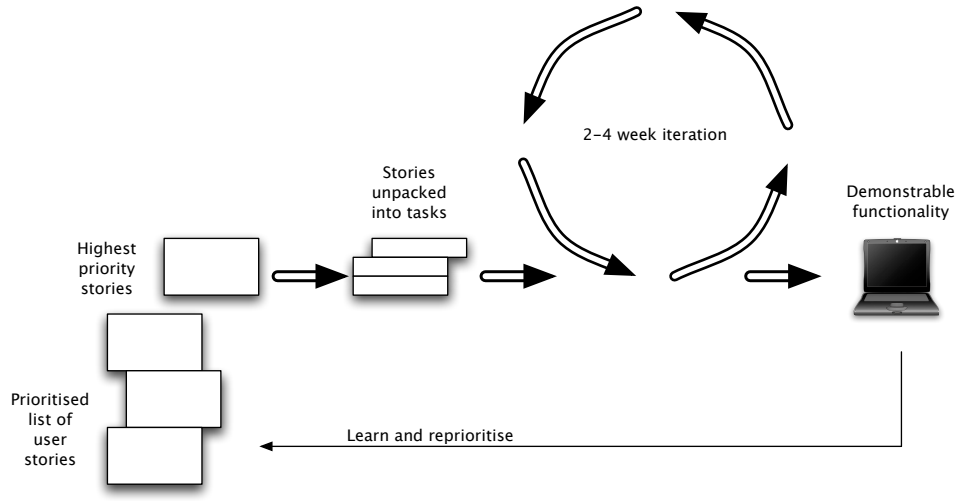
A.1.2 Release planning

The aim of release planning is to agree the scope of the project: which user stories will be implemented in which software release. Releases should be made early and often to maximise feedback. In contrast to plan-driven development, the release plan can be altered throughout development as new user stories arise, old ones are reprioritised or experience in certain tasks is gained.

A.1.3 Iteration planning

In XP, work is organised around iterations that ideally last two to four weeks. The user stories to be implemented for the iteration are chosen from the release plan and are unpacked into finer-grained programming tasks. This helps to estimate the effort required for each user story and makes sure that only the right amount is assigned to a particular iteration. Figure A.2 illustrates the process.

Figure A.2 The iterative nature of XP. Image originally from [Pitt-Francis et al., 2008].



A.1.4 Test-driven development

Test-driven development requires that a unit test, defining the requirements of the code, is written before each aspect of the code, so that it will initially fail. Then code will be added incrementally until the test passes. Designing and implementing unit tests is the XP answer to the design stage of waterfall-like development methodologies. Only the cases covered by the unit test will be considered when coding, preventing over-design.

A.1.5 Refactoring

Test-driven development is not only a testing methodology but also a design methodology, when combined with refactoring. The lack of an overall design (as opposed to unit design) can lead to situations where similar code is repeated in several places. This is considered bad coding practice. Refactoring removes this duplication and keeps the design clean. Refactoring cannot prevent this situation from happening, but can address it.

A.1.6 Collective code ownership

Collective code ownership encourages everyone to contribute new ideas to all segments of the project. Any developer can change any line of code to add functionality, fix bugs or refactor. No individual becomes a bottleneck for changes.

A.1.7 Pair programming

Pair programming refers to the practice of two people sitting at one keyboard and screen when programming. While one person is typing, the other is reviewing their work and thinking strategically. The pair alternates who is typing. This form of programming has two major impacts in code quality: i) code is reviewed on the fly, so coding standards and proper documentation are ensured, ii) bugs produced by variables being mixed up or wrong loop guards and conditionals can be easily spotted. Pair programming is also combined with pair rotation: from time to time one of the developers in a pair will swap with a developer from a different pair. Frequent rotations ensure knowledge about new functionality spreads quickly. Pair programming plays a fundamental role in the training of new developers.

A.1.8 Continuous integration

When code is developed by multiple concurrent streams of work, pairs have local copies of the repository where new tests/functionality is added. Integration is the process whereby this new functionality is committed to the repository. After every commit the whole set of tests in the system (old and new ones) is run to ensure there is no regression in the functionality. XP advocates frequent integration (every few hours) in order to minimise merging problems.

A.1.9 Stand-up meetings

A stand-up meeting takes place between the whole team every day. The meeting is held with the team members standing up, in order to keep the meeting short. People report what they have done since the last stand-up meeting and what they plan to do before the next one. It is also a chance to raise any problems that are holding up progress. The stand-up meeting allows each member to have his or her say in steering the project.

A.1.10 Whole team

This practice refers to the use of a cross-functional team with all the skills and perspectives necessary for the project to succeed. Face-to-face communication replaces the creation and maintenance of complex methodology and requirements documents. Therefore, it is advisable for the team to be located at the same institution. Small teams of 2–12 people are reported to work better [Beck and Andres, 2004].

A.1.11 Further details

Please refer to [Pitt-Francis et al., 2008] for more details on agile methods and how they were adapted to the characteristics of the Chaste project. Refinements of the original methodology were required to accommodate it to the reality of academia and our workforce.

Bibliography

Ashihara, T., Constantino, J., and Trayanova, N. A. (2008). Tunnel propagation of postshock activations as a hypothesis for fibrillation induction and isoelectric window. *Circulation Research*, 102(6):737–745.

Baker, A., Falgout, R., Kolev, T., and Yang, U. (2010). Multigrid smoothers for ultra-parallel computing. Technical Report LLNL-JRNL-435315, Lawrence Livermore National Laboratory.

Barrett, R., Berry, M., Chan, T. F., Demmel, J., Donato, J., Dongarra, J., Eijkhout, V., Pozo, R., Romine, C., and der Vorst, H. V. (1994). *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*. SIAM, Philadelphia, PA, 2nd edition.

Beck, K. and Andres, C. (2004). *Extreme Programming Explained : Embrace Change*. Addison-Wesley Professional, 2nd edition.

Belhamadia, Y. (2008). A time-dependent adaptive remeshing for electrical waves of the heart. *Biomedical Engineering, IEEE Transactions on*, 55(2):443 –452.

Benzi, M. (2002). Preconditioning techniques for large linear systems: A survey. *Journal of Computational Physics*, 182(2):418–477.

Bernabeu, M. O., Bishop, M. J., Pitt-Francis, J., Gavaghan, D., Grau, V., and Rodriguez, B. (2008). High performance computer simulations for the study of

biological function in 3d heart models incorporating fibre orientation and realistic geometry at para-cellular resolution. In *Computers in Cardiology, 2008*, pages 721–724.

Bernabeu, M. O., Corrias, A., Pitt-Francis, J., Rodriguez, B., Bethwaite, B., Enticott, C., Garic, S., Peachey, T., Tan, J., Abramson, D., and Gavaghan, D. J. (2009). Grid computing simulations of ion channel block effects on the ECG using 3D anatomically-based models. In *Computers in Cardiology, 2009*, pages 213–216.

Bernabeu, M. O. and Kay, D. (2011). Scalable parallel preconditioners for an open source cardiac electrophysiology simulation package. *Procedia Computer Science*, 4:821–830. Proceedings of the International Conference on Computational Science, ICCS 2011.

Bernabeu, M. O., Pathmanathan, P., Pitt-Francis, J., and Kay, D. (2010a). Stimulus protocol determines the most computationally efficient preconditioner for the bidomain equations. *IEEE Transactions on Biomedical Engineering*, 57(12):2806 –2815.

Bernabeu, M. O., Wallman, M., and Rodriguez, B. (2010b). Shock-induced arrhythmogenesis in the human heart: A computational modelling study. In *Engineering in Medicine and Biology Society (EMBC), 2010 Annual International Conference of the IEEE*, pages 760–763.

Bishop, M. J. (2008). *Optical Mapping Signal Synthesis*. PhD thesis, Computing Laboratory, University of Oxford.

Bishop, M. J. and Plank, G. (2011). Representing cardiac bidomain bath-loading effects by an augmented monodomain approach: Application to complex ventricular models. *IEEE Transactions on Biomedical Engineering*, 58(4):1066–1075.

Bishop, M. J., Plank, G., Burton, R. A., Schneider, J. E., Gavaghan, D. J., Grau, V., and Kohl, P. (2009). Development of an Anatomically-Detailed MRI-Derived Rabbit Ventricular Model and Assessment of its Impact on Simulation of Electrophysiological Function. *Am J Physiol Heart Circ Physiol*, 298(2):H699–718.

Bordas, R., Carpentieri, B., Fotia, G., Maggio, F., Nobes, R., Pitt-Francis, J., and Southern, J. (2009). Simulation of cardiac electrophysiology on next-generation high-performance computers. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 367(1895):1951–1969.

Botsch, M., Bommes, D., Vogel, C., and Kobbelt, L. (2004). Gpu-based tolerance volumes for mesh processing. *Pacific Conference on Computer Graphics and Applications*, 0:237–243.

Burton, R., Plank, G., Schneider, J., Grau, V., Ahammer, H., Keeling, S., Lee, J., Smith, N., Gavaghan, D., Trayanova, N., and Kohl, P. (2006). 3-d models of individual cardiac histo-anatomy: tools and challenges. *Ann NY Acad Sci*, 1380:301–319.

Cai, X. and Lines, G. T. (2002). Enabling numerical and software technologies for studying the electrical activity in human heart. In *Proceedings of the 6th International Conference on Applied Parallel Computing Advanced Scientific Computing*, PARA '02, pages 3–17, London, UK. Springer-Verlag.

Calvetti, D., Golub, G., and Reichel, L. (1994). An adaptive chebyshev iterative method for nonsymmetric linear systems based on modified moments. *Numerische Mathematik*, 67:21–40.

Calvetti, D. and Reichel, L. (1996). A hybrid iterative method for symmetric positive definite linear systems. *Numerical Algorithms*, 11:79–98.

Chen, P.-S., Shibata, N., Dixon, E., Wolf, P., Danieley, N., Sweeney, M., Smith, W., and Ideker, R. (1986). Activation during ventricular defibrillation in open-chest dogs: evidence of complete cessation and regeneration of ventricular fibrillation after unsuccessful shocks. *J Clin Invest*, 77(3):810–823.

Cherry, E. M., Greenside, H. S., and Henriquez, C. S. (2003). Efficient simulation of three-dimensional anisotropic cardiac tissue using an adaptive mesh refinement method. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 13(3):853–865.

Clayton, R., Bernus, O., Cherry, E., Dierckx, H., Fenton, F., Mirabella, L., Panfilov, A., Sachse, F., Seemann, G., and Zhang, H. (2011). Models of cardiac tissue electrophysiology: Progress, challenges and open questions. *Progress in Biophysics and Molecular Biology*, 104(1-3):22–48.

Clerc, L. (1976). Directional differences of impulse spread in trabecular muscle from mammalian heart. *The Journal of Physiology*, 255(2):335–346.

Colli-Franzone, P. and Pavarino, L. F. (2004). A parallel solver for reaction-diffusion systems in computational electrocardiology. *Mathematical Models and Methods in Applied Sciences*, 14(6):883–912.

Concus, P., Golub, G. H., and O'Leary, D. P. (1976). A generalized conjugate gradient method for the numerical solution of elliptic partial differential equations. Technical report, Stanford University, Stanford, CA, USA.

Cooper, J., McKeever, S., and Garny, A. (2006). On the application of partial evaluation to the optimisation of cardiac electrophysiological simulations. In *Proceedings of the 2006 ACM SIGPLAN symposium on Partial evaluation and semantics-based program manipulation*, PEPM '06, pages 12–20, New York, NY, USA.

Corrias, A., Jie, X., Romero, L., Bishop, M. J., Bernabeu, M., Pueyo, E., and Rodriguez, B. (2010). Arrhythmic risk biomarkers for the assessment of drug cardiotoxicity: from experiments to computer simulations. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 368(1921):3001–3025.

Corrias, A., Pathmanathan, P., Gavaghan, D. J., and Buist, M. L. (2011). Modelling tissue electrophysiology with multiple cell types: applications of the extended bidomain framework. Submitted to Progress in Biophysics and Molecular Biology.

Coudiere, Y. and Pierre, C. (2006). Stability and convergence of a finite volume method for two systems of reaction-diffusion equations in electro-cardiology. *Nonlinear Analysis: Real World Applications*, 7:916–935.

Cuellar, A. A., Lloyd, C. M., Nielsen, P. F., Bullivant, D. P., Nickerson, D. P., and Hunter, P. J. (2003). An overview of cellml 1.1, a biological model description language. *SIMULATION*, 79(12):740–747.

DeBruin, K. A. and Krassowska, W. (1999). Modeling electroporation in a single cell. i. effects of field strength and rest potential. *Biophysical Journal*, 77(3):1213–1224.

Demir, S. S., Clark, J. W., Murphey, C. R., and Giles, W. R. (1994). A mathematical model of a rabbit sinoatrial node cell. *American Journal of Physiology - Cell Physiology*, 266(3):C832–C852.

Deuflhard, P., Erdmann, B., Roitzsch, R., and Lines, G. (2009). Adaptive finite element simulation of ventricular fibrillation dynamics. *Computing and Visualization in Science*, 12:201–205. 10.1007/s00791-008-0088-y.

DiFrancesco, D. and Noble, D. (1985). A model of cardiac electrical activity incor-

porating ionic pumps and concentration changes. *Philosophical Transactions of the Royal Society of London B: Biological Sciences*, 307(1133):353–398.

dos Santos, R., Plank, G., Bauer, S., and Vigmond, E. (2004). Parallel multigrid preconditioner for the cardiac bidomain model. *IEEE Transactions on Biomedical Engineering*, 51(11):1960 –1968.

Efimov, I. R., Cheng, Y., Van Wagoner, D. R., Mazgalev, T., and Tchou, P. J. (1998). Virtual electrode-induced phase singularity: A basic mechanism of defibrillation failure. *Circ Res*, 82(8):918–925.

Elharrar, V. and Zipes, D. (1977). Cardiac electrophysiologic alterations during myocardial ischemia. *The American Journal of Physiology*, 233(3):H329–345.

Elman, H., Silvester, D., and Wathen, A. (2005). *Finite elements and fast iterative solvers: with applications in incompressible fluid dynamics*. Numerical mathematics and scientific computation. Oxford University Press.

Fabbri, R., Luciano, Torelli, J. C., and Bruno, O. M. (2008). 2d euclidean distance transform algorithms: A comparative survey. *ACM Comput. Surv.*, 40(1):1–44.

Falgout, R. (2006). An introduction to algebraic multigrid computing. *Computing in Science Engineering*, 8(6):24 –33.

Fishler, M. G. and Thakor, N. V. (1991). A massively parallel computer model of propagation through a two-dimensional cardiac syncytium. *Pacing and Clinical Electrophysiology*, 14(11):1694–1699.

Fowler, M. (2000). Put your process on a diet. *Software Development*, 8(12):32–36.

Golub, G. H. and Van Loan, C. F. (1996). *Matrix Computations*. Johns Hopkins Studies in Mathematical Sciences. The Johns Hopkins University Press, 3rd edition.

Gutknecht, M. H. and Röllin, S. (2002). The chebyshev iteration revisited. *Parallel Comput.*, 28:263–283.

Henriquez, C. S. (1993). Simulating the electrical behavior of cardiac tissue using the bidomain model. *Crit Rev Biomed Eng*, 21(1):1–77.

Hooks, D. A., Tomlinson, K. A., Marsden, S. G., LeGrice, I. J., Smaill, B. H., Pullan, A. J., and Hunter, P. J. (2002). Cardiac microstructure: Implications for electrical propagation and defibrillation in the heart. *Circulation Research*, 91(4):331–338.

Hooks, D. A., Trew, M. L., Caldwell, B. J., Sands, G. B., LeGrice, I. J., and Smaill, B. H. (2007). Laminar arrangement of ventricular myocytes influences electrical behavior of the heart. *Circ Res*, 101:e103–e112.

Irons, B. M. (1970). A frontal solution program for finite element analysis. *International Journal for Numerical Methods in Engineering*, 2(1):5–32.

Jacquemet, V. and Henriquez, C. (2005). Finite volume stiffness matrix for solving anisotropic cardiac propagation in 2-d and 3-d unstructured meshes. *IEEE Transactions on Biomedical Engineering*, 52(8):1490 –1492.

Jalife, J. (2000). Ventricular fibrillation: Mechanisms of initiation and maintenance. *Annual Review of Physiology*, 62:25–50.

Johnson, C. (1987). *Numerical solution of partial differential equations by the finite element method*. Cambridge University Press.

Jones, M. W., Baerentzen, J. A., and Sramek, M. (2006). 3d distance fields: a survey of techniques and applications. *IEEE Transactions on Visualization and Computer Graphics*, 12(4):581–599.

Karypis, G. and Kumar, V. (1998). A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on Scientific Computing*, 20:359–392.

Keener, J. and Sneyd, J. (1998). *Mathematical Physiology*. Springer, 1st edition.

Kerckhoffs, R., Healy, S., Usyk, T., and McCulloch, A. (2006). Computational methods for cardiac electromechanics. *Proceedings of the IEEE*, 94(4):769 –783.

Legrice, I. J., Hunter, P. J., and Smaill, B. H. (1997). Laminar structure of the heart: a mathematical model. *The American journal of physiology*, 272:H2466–H2476.

LeGrice, I. J., Smaill, B. H., Chai, L. Z., Edgar, S. G., Gavin, J. B., and Hunter, P. J. (1995). Laminar structure of the heart: ventricular myocyte arrangement and connective tissue architecture in the dog. *American Journal of Physiology - Heart and Circulatory Physiology*, 269(2):H571–H582.

Linge, S., Sundnes, J., Hanslien, M., Lines, G., and Tveito, A. (2009). Numerical solution of the bidomain equations. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 367(1895):1931–1950.

Luo, C.-H. and Rudy, Y. (1991). A model of the ventricular cardiac action potential - depolarisation, repolarisation and their interaction. *Circ Res*, 68:1501–1526.

Mahajan, A., Shiferaw, Y., Sato, D., Baher, A., Olcese, R., Xie, L.-H., Yang, M.-J., Chen, P.-S., Restrepo, J. G., Karma, A., Garfinkel, A., Qu, Z., and Weiss, J. N. (2008). A rabbit ventricular action potential model replicating cardiac dynamics at rapid heart rates. *Biophysical Journal*, 94(2):392 – 410.

Maharaj, T., Blake, R., Trayanova, N., Gavaghan, D., and Rodriguez, B. (2008). The role of transmural ventricular heterogeneities in cardiac vulnerability to electric shocks. *Progress in Biophysics and Molecular Biology*, 96(1-3):321 – 338.

Maleckar, M. M., Greenstein, J. L., Trayanova, N. A., and Giles, W. R. (2008). Mathematical simulations of ligand-gated and specific cell-type effects in the human atrium. *Progress in Biophysics and Molecular Biology*, 98(2-3):161–70.

Manteuffel, T. A. (1977). The tchebychev iteration for nonsymmetric linear systems. *Numerische Mathematik*, 28:307–327.

Munteanu, M. and Pavarino, L. F. (2009). Decoupled schwarz algorithms for implicit discretizations of nonlinear monodomain and bidomain systems. *Mathematical Models and Methods in Applied Sciences*, 19(7):1065–1097.

Murillo, M. and Cai, X.-C. (2004). A fully implicit parallel algorithm for simulating the non-linear electrical activity of the heart. *Numerical Linear Algebra with Applications*, 11(2-3):261–277.

Ng, K. T., Hutchinson, S. A., and Gao, S. (1995). Numerical analysis of electrical defibrillation: The parallel approach. *Journal of Electrocardiology*, 28(Supplement 1):15–20.

Niederer, S., Mitchell, L., Smith, N., and Plank, G. (2011a). Simulating human cardiac electrophysiology on clinical time-scales. *Frontiers in Physiology*, 2(0).

Niederer, S. A., Kerfoot, E., Benson, A. P., Bernabeu, M. O., Bernus, O., Bradley, C., Cherry, E. M., Clayton, R., Fenton, F. H., Garny, A., Heidenreich, E., Land, S., Maleckar, M., Pathmanathan, P., Plank, G., Rodriguez, J. F., Roy, I., Sachse, F. B., Seemann, G., Skavhaug, O., and Smith, N. P. (2011b). Verification of cardiac tissue electrophysiology simulators using an N-version benchmark. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 369(1954):4331–4351.

Nielsen, P. M., Le Grice, I. J., Smaill, B. H., and Hunter, P. J. (1991). Mathematical model of geometry and fibrous structure of the heart. *American Journal of Physiology - Heart and Circulatory Physiology*, 260(4):H1365–H1378.

Noble, D. (1960). Cardiac action and pacemaker potentials based on the hodgkin-huxley equations. *Nature*, 188:495–497.

Nygren, A., Fiset, C., Firek, L., Clark, J. W., Lindblad, D. S., Clark, R. B., and Giles, W. R. (1998). Mathematical model of an adult human atrial cell : The role of k+ currents in repolarization. *Circ Res*, 82(1):63–81.

Osborne, J. (2009). *Numerical and Computational Methods for Simulating Multiphase Models of Tissue Growth*. PhD thesis, Computing Laboratory, University of Oxford.

Panfilov, A. V. (1998). Spiral breakup as a model of ventricular fibrillation. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 8(1):57–64.

Pathmanathan, P., Bernabeu, M. O., Bordas, R., Cooper, J., Garny, A., Pitt-Francis, J. M., Whiteley, J. P., and Gavaghan, D. J. (2010). A numerical guide to the solution of the bidomain equations of cardiac electrophysiology. *Progress in Biophysics and Molecular Biology*, 102(2-3):136–155.

Pavarino, L. F. and Scacchi, S. (2008). Multilevel additive schwarz preconditioners for the bidomain reaction-diffusion system. *SIAM J. Sci. Comput.*, 31(1):420–443.

Penland, R. C., Harrild, D. M., and Henriquez, C. S. (2002). Modeling impulse propagation and extracellular potential distributions in anisotropic cardiac tissue using a finite volume element discretization. *Computing and Visualization in Science*, 4:215–226.

Pennacchio, M. and Simoncini, V. (2002). Efficient algebraic solution of reaction-diffusion systems for the cardiac excitation process. *J. Comput. Appl. Math.*, 145(1):49–70.

Pennacchio, M. and Simoncini, V. (2009). Algebraic multigrid preconditioners for the bidomain reaction–diffusion system. *Appl. Numer. Math.*, 59(12):3033–3050.

Piper, J. and Granum, E. (1987). Computing distance transformations in convex and non-convex domains. *Pattern Recognition*, 20(6):599 – 615.

Pitt-Francis, J., Bernabeu, M. O., Cooper, J., Garny, A., Momtahan, L., Osborne, J., Pathmanathan, P., Rodriguez, B., Whiteley, J. P., and Gavaghan, D. J. (2008). Chaste: Using agile programming techniques to develop computational biology software. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 366(1878):3111–3136+.

Pitt-Francis, J., Pathmanathan, P., Bernabeu, M. O., Bordas, R., Cooper, J., Fletcher, A. G., Mirams, G. R., Murray, P., Osborne, J. M., Walter, A., Chapman, S. J., Garny, A., van Leeuwen, I. M., Maini, P. K., Rodríguez, B., Waters, S. L., Whiteley, J. P., Byrne, H. M., and Gavaghan, D. J. (2009). Chaste: A test-driven approach to software development for biological modelling. *Computer Physics Communications*, 180(12):2452–2471.

Plank, G., Liebmann, M., dos Santos, R. W., Vigmond, E. J., and Haase, G. (2007). Algebraic multigrid preconditioner for the cardiac bidomain model. *IEEE Transactions on Biomedical Engineering*, 54:585–596.

Plank, G., Prassl, A., Hofer, E., and Trayanova, N. (2008). Evaluating intramural virtual electrodes in the myocardial wedge preparation: Simulations of experimental conditions. *Biophysical Journal*, 94(5):1904 – 1915.

Pogwizd, S. and Corr, P. (1990). Mechanisms underlying the development of ventricular fibrillation during early myocardial ischemia. *Circulation Research*, 66(3):672–695.

Pollard, A. and Barr, R. (1991). Computer simulations of activation in an anatomically based model of the human ventricular conduction system. *IEEE Transactions on Biomedical Engineering*, 38(10):982 –996.

Porras, D., Rogers, J., Smith, W., and Pollard, A. (2000). Distributed computing for

membrane-based modeling of action potential propagation. *IEEE Transactions on Biomedical Engineering*, 47(8):1051 –1057.

Potse, M., Dubé, B., Richer, J., Vinet, A., and Gulrajani, R. M. (2006). A comparison of monodomain and bidomain reaction-diffusion models for action potential propagation in the human heart. *IEEE Transactions on Biomedical Engineering*, 53(12 Pt 1):2425–2435.

Pullan, A., Cheng, L., and Buist, M. (2005). *Mathematically modelling the electrical activity of the heart: from cell to body surface and back again*. World Scientific.

Quan, W., Evans, S., and Hastings, H. (1998). Efficient integration of a realistic two-dimensional cardiac tissue model by domain decomposition. *IEEE Transactions on Biomedical Engineering*, 45(3):372 –385.

Ragnemalm, I. (1992). Neighborhoods for distance transformations using ordered propagation. *CVGIP: Image Understanding*, 56:399–409.

Reumann, M., Fitch, B., Rayshubskiy, A., Keller, D., Seemann, G., Dossel, O., Pitman, M., and Rice, J. (2009). Strong scaling and speedup to 16,384 processors in cardiac electro-mechanical simulations. In *Engineering in Medicine and Biology Society, 2009. EMBC 2009. Annual International Conference of the IEEE*, pages 2795–2798.

Rodríguez, B., Burrage, K., Gavaghan, D., Grau, V., Kohl, P., and Noble, D. (2010). The systems biology approach to drug development: Application to toxicity assessment of cardiac drugs. *Clin Pharmacol Ther*, 88(1):130–134.

Rodríguez, B., Li, L., Eason, J., Efimov, I., and Trayanova, N. (2005). Differences between left and right ventricular chamber geometry affect cardiac vulnerability to electric shocks. *Circ Res*, 97(2):168–75.

Rodríguez, B., Tice, B. M., Eason, J. C., Aguel, F., and Trayanova, N. (2004). Cardiac vulnerability to electric shocks during phase 1a of acute global ischemia. *Heart Rhythm*, 1(6):695 – 703.

Rodríguez, B. and Trayanova, N. (2003). Upper limit of vulnerability in a defibrillation model of the rabbit ventricles. *Journal of Electrocardiology*, 36(Supplement 1):51 – 56.

Rogers, J. (2002). Wave front fragmentation due to ventricular geometry in a model of the rabbit heart. *Chaos*, 12(3):779–787.

Rosenfeld, A. and Pfaltz, J. L. (1966). Sequential operations in digital picture processing. *J. ACM*, 13(4):471–494.

Roth, B. (1995). A mathematical model of make and break electrical stimulation of cardiac tissue by a unipolar anode or cathode. *IEEE Transactions on Biomedical Engineering*, 42(12):1174 –1184.

Roth, B. J. and Krassowska, W. (1998). The induction of reentry in cardiac tissue. the missing link: How electric fields alter transmembrane potential. *Chaos*, 8(1):204–220.

Saleheen, H., Claessen, P., and Ng, K. (1997). Three-dimensional finite-difference bidomain modeling of homogeneous cardiac tissue on a data-parallel computer. *IEEE Transactions on Biomedical Engineering*, 44(2):200 –204.

Sampson, K. J. and Henriquez, C. S. (2005). Electrotonic influences on action potential duration dispersion in small hearts: a simulation study. *Am J Physiol Heart Circ Physiol*, 289(1):H350–360.

Scacchi, S. and Pavarino, L. F. (2008). Multilevel schwarz and multigrid preconditioners for the bidomain system. In *Domain Decomposition Methods in Science*

and Engineering XVII, volume 60 of *Lecture Notes in Computational Science and Engineering*, pages 631–638. Springer Berlin Heidelberg.

Schmitt, O. H. (1969). Biological information processing using the concept of interpenetrating domains. In Leibovic, K. N., editor, *Information processing in the nervous system*, page 325. Springer-Verlag.

Schroll, H. J., Lines, G. T., and Tveito, A. (2007). On the accuracy of operator splitting for the monodomain model of electrophysiology. *International Journal of Computer Mathematics*, 84(6):871–885.

Sebastian, R., Ordas, S., Plank, G., Rodriguez, B., Vigmond, E. J., and Frangi, A. F. (2008). Assessing influence of conductivity in heart modelling with the aim of studying cardiovascular diseases. In Hu, X. P. and Clough, A. V., editors, *Medical Imaging 2008: Physiology, Function, and Structure from Medical Images*, volume 6916, page 691627. SPIE.

Sepulveda, N., Roth, B., and Wikswo, Jr, J. (1989). Current injection into a two-dimensional anisotropic bidomain. *Biophysical Journal*, 55(5):987 – 999.

Shewchuk, J. R. (1994). An introduction to the conjugate gradient method without the agonizing pain. Technical report, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA.

Silvester, D., Elman, H., Kay, D., and Wathen, A. (2001). Efficient preconditioning of the linearized navier–stokes equations for incompressible flow. *J. Comput. Appl. Math.*, 128:261–279.

Skouibine, K., Trayanova, N., and Moore, P. (2000). Success and failure of the defibrillation shock: Insights from a simulation study. *Journal of Cardiovascular Electrophysiology*, 11(7):785–795.

Southern, J., Gorman, G., Piggott, M., and Farrell, P. (2011). Parallel anisotropic mesh adaptivity with dynamic load balancing for cardiac electrophysiology. *Journal of Computational Science*, (0):–.

Southern, J., Gorman, G., Piggott, M., Farrell, P., Bernabeu, M., and Pitt-Francis, J. (2010). Simulating cardiac electrophysiology using anisotropic mesh adaptivity. *Journal of Computational Science*, 1(2):82 – 88.

Southern, J., Plank, G., Vigmond, E., and Whiteley, J. (2009). Solving the coupled system improves computational efficiency of the bidomain equations. *IEEE Transactions on Biomedical Engineering*, 56(10):2404–2412.

Streeter, Jr., D., Spotnitz, H., Patel, D., Ross, Jr., J., and Sonnenblick, E. (1969). Fiber orientation in the canine left ventricle during diastole and systole. *Circ Res*, 24:339–347.

Sud, A., Otaduy, M. A., and Manocha, D. (2004). DiFi: Fast 3D distance field computation using graphics hardware. In *Computer Graphics Forum*, volume 23 (3), pages 557–566.

Süli, E. (2007). *Finite Element methods for Partial Differential Equations*. Oxford University Computing Laboratory, Lecture Notes.

Sundnes, J., Artebrant, R., Skavhaug, O., and Tveito, A. (2009). A second-order algorithm for solving dynamic cell membrane equations. *IEEE Transactions on Biomedical Engineering*, 56(10):2546 –2548.

Sundnes, J., Lines, G. T., and Tveito, A. (2005). An operator splitting method for solving the bidomain equations coupled to a volume conductor model for the torso. *Mathematical Biosciences*, 194(2):233 – 248.

Sundnes, J., Nielsen, B. F., Mardal, K. A., Cai, X., Lines, G. T., and Tveito, A.

(2006). On the computational complexity of the bidomain and the monodomain models of electrophysiology. *Annals of biomedical engineering*, 34(7):1088–1097.

Tallent, N. R., Mellor-Crummey, J. M., Adhianto, L., Fagan, M. W., and Krentel, M. (2009). Diagnosing performance bottlenecks in emerging petascale applications. In *Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis*, SC '09, pages 51:1–51:11, New York, NY, USA. ACM.

ten Tusscher, K. H., Hren, R., and Panfilov, A. V. (2007). Organization of ventricular fibrillation in the human heart. *Circ Res*, 100(12):e87–101.

ten Tusscher, K. H. W. J. and Panfilov, A. V. (2006). Alternans and spiral breakup in a human ventricular tissue model. *American Journal of Physiology - Heart and Circulatory Physiology*, 291(3):H1088–H1100.

Trangenstein, J. A. and Kim, C. (2004). Operator splitting and adaptive mesh refinement for the luo-rudy i model. *J. Comput. Phys.*, 196:645–679.

Trayanova, N. (2001). Concepts of ventricular defibrillation. *Philosophical Transactions of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, 359(1783):1327–1337.

Trayanova, N., Eason, J., and Aguel, F. (2002). Computer simulations of cardiac defibrillation: a look inside the heart. *Computing and Visualization in Science*, 4(4):259–270.

Trayanova, N., Plank, G., and Rodríguez, B. (2006). What have we learned from mathematical models of defibrillation and postshock arrhythmogenesis? application of bidomain simulations. *Heart Rhythm*, 3(10):1232 – 1235.

Trayanova, N. A., Skouibine, K. B., and Aguel, F. (1998). The role of cardiac tissue structure in defibrillation. *Chaos*, 8(1):221–233.

Trayanova, N. A. and Tice, B. M. (2009). Integrative computational models of cardiac arrhythmias - simulating the structurally realistic heart. *Drug Discovery Today: Disease Models*, 6(3):85–91.

Trew, M., Ashton, J., Caldwell, B., and Smaill, B. (2009). Shock induced electrical activation in structurally detailed models of pig left-ventricular tissue. In *Engineering in Medicine and Biology Society, 2009. EMBC 2009. Annual International Conference of the IEEE*, pages 3948 –3951.

Tung, L. (1978). *A bi-domain model for describing ischemic myocardial d-c potentials*. PhD thesis, Massachusetts Institute of Technology.

Usyk, T. P. and McCulloch, A. D. (2003). Relationship between regional shortening and asynchronous electrical activation in a three-dimensional model of ventricular electromechanics. *J Cardiovasc Electrophysiol*, 14:S196–202.

van der Vorst, H. A. (2003). *Iterative Krylov Methods for Large Linear Systems*. Cambridge University Press, Cambridge.

Vázquez, M., Arís, R., Houzeaux, G., Aubry, R., Villar, P., Garcia-Barnés, J., Gil, D., and Carreras, F. (2011). A massively parallel computational electrophysiology model of the heart. *International Journal for Numerical Methods in Biomedical Engineering*, (online).

Vetter, F. J. and McCulloch, A. D. (1998). Three-dimensional analysis of regional cardiac function: a model of rabbit ventricular anatomy. *Progress in Biophysics and Molecular Biology*, 69(2-3):157–183.

Vigmond, E. and Clements, C. (2007). Construction of a computer model to investigate sawtooth effects in the purkinje system. *IEEE Transactions on Biomedical Engineering*, 54(3):389–399.

Vigmond, E., dos Santos, R. W., Prassl, A., Deo, M., and Plank, G. (2008). Solvers for the cardiac bidomain equations. *Progress in Biophysics and Molecular Biology*, 96(1-3):3–18.

Vigmond, E., Vadakkumpadan, F., Gurev, V., Arevalo, H., Deo, M., Plank, G., and Trayanova, N. (2009). Towards predictive modelling of the electrophysiology of the heart. *Experimental Physiology*, 94(5):563–577.

Vigmond, E. J., Aguel, F., and Trayanova, N. A. (2002). Computational techniques for solving the bidomain equations in three dimensions. *IEEE Transactions on Biomedical Engineering*, 49(11):1260–1269.

Vigmond, E. J., Hughes, M., Plank, G., and Leon, L. J. (2003). Computational tools for modeling electrical activity in cardiac tissue. *Journal of Electrocardiology*, 36(Supplement 1):69–74.

Wathen, A. (1988). Spectral bounds and preconditioning methods using element-by-element analysis for galerkin finite element equations. In Whiteman, J., editor, *Proc. of MAFELAP(1987)*, pages 157–168. Academic Press, London.

Wathen, A. J. (1987). Realistic Eigenvalue Bounds for the Galerkin Mass Matrix. *IMA Journal of Numerical Analysis*, 7(4):449–457.

Wathen, A. J. (1989). An analysis of some element-by-element techniques. *Computer Methods in Applied Mechanics and Engineering*, 74(3):271–287.

Weidmann, S. (1970). Electrical constants of trabecular muscle from mammalian heart. *The Journal of Physiology*, 210(4):1041–1054.

Weiser, M., Erdmann, B., and Deuflhard, P. (2008). On efficiency and accuracy in cardioelectric simulation. Technical Report 08-41, ZIB, Takustr.7, 14195 Berlin.

Wesseling, P. (1992). *An Introduction to Multigrid Methods*. John Wiley & Sons.

Whiteley, J. (2006). An efficient numerical technique for the solution of the monodomain and bidomain equations. *IEEE Transactions on Biomedical Engineering*, 53:2139–2147.

Wikswo, Jr., J., Lin, S. F., and Abbas, R. A. (1995). Virtual electrodes in cardiac tissue: a common mechanism for anodal and cathodal stimulation. *Biophysical Journal*, 69:2195–2210.

Winslow, R. L., Kimball, A. L., Varghese, A., and Noble, D. (1993). Simulating cardiac sinus and atrial network dynamics on the connection machine. *Physica D: Nonlinear Phenomena*, 64(1-3):281–298.

Winslow, R. L., Rice, J., Jafri, S., Marbán, E., and O'Rourke, B. (1999). Mechanisms of altered excitation-contraction coupling in canine tachycardia-induced heart failure, ii : Model studies. *Circulation Research*, 84(5):571–586.

Wolf, P., Abbott, R., and Kannel, W. (1991). Atrial fibrillation as an independent risk factor for stroke: the framingham study. *Stroke*, 22(8):983–988.

Zemzemi, N., Bernabeu, M. O., Saiz, J., and Rodriguez, B. (2011). Simulating drug-induced effects on the heart: From ion channel to body surface electrocardiogram. In Metaxas, D. N. and Axel, L., editors, *Functional Imaging and Modeling of the Heart*, volume 6666 of *Lecture Notes in Computer Science*, pages 259–266. Springer.

Zhang, H., Holden, A. V., Kodama, I., Honjo, H., Lei, M., Varghese, T., and Boyett, M. R. (2000). Mathematical models of action potentials in the periphery and center of the rabbit sinoatrial node. *American Journal of Physiology - Heart and Circulatory Physiology*, 279(1):H397–H421.