Efficient Simulation of Cardiac Electrical Propagation using Adaptive High-Order Finite Elements

Christopher J. Arthurs

Balliol College

University of Oxford

A thesis submitted for the degree of

Doctor of Philosophy

Michaelmas Term 2012

Abstract

This thesis investigates the high-order hierarchical finite element method, also known as the finite element p-version, as a computationally-efficient technique for generating numerical solutions to the cardiac monodomain equation. We first present it as a uniform-order method, and through an *a priori* error bound we explain why the associated cardiac cell model must be thought of as a PDE and approximated to high-order in order to obtain the accuracy that the p-version is capable of. We perform simulations demonstrating that the achieved error agrees very well with the *a priori* error bound. Further, in terms of solution accuracy for time taken to solve the linear system that arises in the finite element discretisation, it is more efficient than the state-of-the-art piecewise linear finite element method. We show that piecewise linear FEM actually introduces quite significant amounts of error into the numerical approximations, particularly in the direction perpendicular to the cardiac fibres with physiological conductivity values, and that without resorting to extremely fine meshes with elements considerably smaller than 70 μm, we can not use it to obtain high-accuracy solutions. In contrast, the p-version can produce extremely high accuracy solutions on meshes with elements around 300 μm in diameter with these conductivities.

Noting that most of the numerical error is due to under-resolving the wave-front in the transmembrane potential, we also construct an adaptive high-order scheme which controls the error locally in each element by adjusting the finite element polynomial basis degree using an analytically-derived *a posteriori* error estimation procedure. This naturally tracks the location of

the wavefront, concentrating computational effort where it is needed most and increasing computational efficiency. The scheme can be controlled by a user-defined error tolerance parameter, which sets the target error within each element as a proportion of the local magnitude of the solution as measured in the $\mathcal{H}^1$ norm. This numerical scheme is tested on a variety of problems in one, two and three dimensions, and is shown to provide excellent error control properties and to be likely capable of boosting efficiency in cardiac simulation by an order of magnitude.

The thesis amounts to a proof-of-concept of the increased efficiency in solving the linear system using adaptive high-order finite elements when performing single-thread cardiac simulation, and indicates that the performance of the method should be investigated in parallel, where it can also be expected to provide considerable improvement. In general, the selection of a suitable preconditioner is key to ensuring efficiency; we make use of a variety of different possibilities, including one which can be expected to scale very well in parallel, meaning that this is an excellent candidate method for increasing the efficiency of cardiac simulation using high-performance computing facilities.

Contents

Acknowledgements

First and foremost, I would like to thank my supervisors David Kay and Martin Bishop for all the time and effort they have put into making this work possible. They have both been fantastic in advising me on the direction of this project and helping me with the things that have gone wrong. I would also like to thank my Transfer and Confirmation of Status panel David Gavaghan and Kathryn Gillow for their advice which helped to focus this project in its early stages, and Joe Pitt-Francis for providing me with time on his GPU server, making the whole-heart simulation possible. Thanks to the Chaste team for their help in expanding my programming ability, and to to Blanca Rodríguez and Kevin Burrage, in addition to David Kay, for introducing me to the field of computational cardiac electrophysiology via a DTC short project.

For various combinations of discussion, advice, support and general tolerance, thanks to John Walmsley, Raf Bordas, Gary Mirams, Mikael Wallman, Eoin Hyde and Alberto Corrias in particular, and to the whole Computational Biology Group in general for providing a supportive and encouraging working environment.

I'd like to thank my family John, Gillian and Steve for their love and support throughout.

Thanks to the EPSRC and the LSI DTC for making this work possible.

Chapter 1

Introduction

1.1 Motivation

New insight into cardiac function is of great importance to medical science, not least because heart disease is the leading cause of death in the developed world; in the United Kingdom it accounts for more than one in six of all deaths [1]. Increased understanding of the working of the heart in both physiological and pathological conditions will therefore aid the development of new treatments for a variety of cardiac and non-cardiac [2, 3] diseases.

A major hurdle we face is that obtaining high spatial and temporal resolution data on the dynamics of the heart is difficult. In a clinical environment, we must settle for low resolution methods such as magnetic resonance imaging (MRI) or electrocardiograms (ECG). Outside of the clinical setting, more information can be obtained from animal studies, for example by optical mapping [4]. Unfortunately, these only provide incomplete data on a limited subset of the parameters of interest. Computational multiscale simulation provides another tool, allowing the measurement and modification of hundreds of different variables in the whole three-dimensional tissue volume, with the added benefits of procedural simplicity and avoiding the need for animal studies.

Myocardial electrical propagation can be simulated using the monodomain or bido-

main PDEs [5, 6]. The simpler of the two, the monodomain, is the problem *for some* $T > 0$ *and for all* $t \in [0, T]$*, find* $u(x,t)$*,* $w(x,t) = (w_1(x,t), \ldots, w_m(x,t))^T$ *such that*

$$C_m \frac{\partial u}{\partial t} - \frac{1}{\beta} \nabla \cdot (\sigma \nabla u) - I_{ionic}(u, w) = I_{stim}(x,t) \text{ in } \Omega, \tag{1.1a}$$

$$\frac{\partial w}{\partial t} - g(u, w) = 0 \text{ in } \Omega, \tag{1.1b}$$

$$u(x, 0) = u^0(x) \; \forall x \in \Omega, \tag{1.1c}$$

$$\hat{n} \cdot (\sigma \nabla u) = 0 \text{ on } \partial\Omega, \tag{1.1d}$$

$$w(x, 0) = w^0(x) \; \forall x \in \Omega. \tag{1.1e}$$

Here, x is a point in the open l-dimensional myocardial domain Ω, Ω has boundary $\partial\Omega$, $\hat{n}$ is the outward-pointing unit surface normal to $\partial\Omega$, and we have the transmembrane potential $u(x,t)$, g describes how the m non-diffusing cell model state variables w vary in time, initial conditions u^0 and w^0, conductivity tensor $\sigma(x)$ time t, cell membrane capacitance C_m, cell surface area to volume ratio β and current $I_{total}(u, w, x, t) = I_{ionic}(u, w) + I_{stim}(x, t)$, consisting of the transmembrane ionic current $I_{ionic}(u, w)$ as described by the cell model and the stimulus current $I_{stim}(x, t)$ as determined by the experimental protocol. Due to its capacity to represent complex geometries with ease, approximations are often obtained using the finite element method (FEM) to discretise the partial differential equations (PDEs) in space on realistic cardiac geometry meshes; this typically results in very large (tens of millions of degrees of freedom (DOF) for human heart geometries) systems of linear equations which must be solved many thousands of times over the course of even a short simulation. Thus, they are extremely computationally demanding, presenting taxing problems even to high-end supercomputing resources; with 16,384 CPU cores, a human heart-beat simulation remains 240 times slower than real-time [7], although at the resolutions used in this simulation the errors would be significant when using realistic tissue conductivities. Alternatively, a higher-resolution rabbit heart mesh [8] can be run on 2,048 cores at a speed around 5,400 times slower than real-time [9], or a recent paper presents simulation on a high-resolution human heart

mesh which is predicted to achieve 8-10 heart-beats per minute; this is impressive, but still seven times slower than real-time, despite making use of 1,572,864 cores of the world's fastest supercomputing resource [10, 11]. Because potential research applications include simulating the progression of conditions such as ischemia on the time-scale of hours, or performing studies involving many hundreds of varied-parameter *in silico* simulations of cardiac tissue drug interaction, and because there is interest in moving to ever more advanced models of cardiac tissue [12], considerable computational efficiency improvements are necessary to produce the fast, accurate results that researchers and clinicians need.

1.2 Thesis Goal

The computational demand associated with mono- or bidomain simulation means that much effort has been invested in developing efficient solution techniques, including work on preconditioning, parallelisation and adaptivity in space and time [13, 14, 15, 16, 17, 18]. In this thesis, we investigate the potential of reducing the number of DOF by using a high-order polynomial FEM [19, 20, 21] to approximate the monodomain PDE in space, with the goal of significantly improving simulation efficiency over the piecewise-linear FEM approach commonly used in the field [22, 23, 24, 25]. For schemes where the polynomial degree p of the elements is adjusted according to the error in the approximation, this is known as the finite element p-version. In the work presented here, we work initially with methods which keep p fixed, and then design and evaluate an adaptive p-version scheme which automatically selects higher order elements locally in the domain where the error would otherwise be large, and opts for lower order elements, which are computationally cheaper, elsewhere. This adaptivity is autonomous, and is driven by an analytically-derived error indicator which provides us with confidence that the error in the solution is being effectively controlled.

Because the *a priori* L^2 error $\|u - u_{hp}\|_{0,2}$ between the true solution u and finite element approximation u_{hp} of spatially semi-discrete parabolic PDEs satisfies

$$\|u - u_{hp}\|_{0,2} \leq Ch^{\mu}p^{-k}\|u\|_{k,2}, \tag{1.2}$$

on a mesh with quasiuniform element diameter h, for some constant C with $\mu = min\{k, p+1\}$ (when u has k square-integrable derivatives, so that $\|u\|_{k,2}$ is finite; see Chapter 4 for more details and for the definition of the norm $\|\cdot\|_{k,2}$) [21], and because for our problem we believe that k is large, we have good reason to look for greater computational efficiency from using larger values of p and h in order to obtain a desired accuracy. This allows for an exponential rate of error reduction as p increases. It is reasonable that we should expect k to be large given that our problem is a homogenised model of a physical process. Careful choice of h and p can result in a linear system with fewer DOF and thus improved computational efficiency. In other fields such as acoustics, elastodynamics and electromagnetics, this approach has been shown to produce a speed-up of five to ten times over a standard linear FEM approach [26].

The motivation for investigating adaptivity is that solutions to System (1.1) take the form of a travelling wave of electrical potential difference; they have a very steep interface at the wavefront, and are relatively flat elsewhere. The wavefront constitutes the part of the solution which is the most challenging to capture accurately, so without adaptivity this will dictate the approximation power of the numerical method that must be used everywhere in the domain. The use of adaptivity deals with this problem by allowing the approximation power to be set locally in the domain, so that the presence of the wavefront does not force the inefficient use of high-power approximation elsewhere.

Work to-date on higher-order elements [27, 28] has focused on hexahedral meshes and what is effectively lumping of the mass matrix [29] (despite claims that an advantage of such high-order methods is that they avoid mass-lumping [26]). The existing approaches demonstrate a two- to three-fold speed up over linear FEM for a 3D parabolic test problem on a coarse cardiac geometry [28]. Our approach allows the use of tetrahedral meshes and lends itself to spatial adaptivity, which we will thoroughly evaluate.

In this thesis, we focus on the monodomain equation, although the presented techniques can easily be extended to the bidomain problem. In one spatial dimension, we provide comparisons of the error in the simulations using different polynomial degrees with theoretical *a priori* results in certain norms, displaying strong agreement. We also use simpler error measures such as the activation time $A(x)$ at a point x, which is defined to be the time t_i for which $u(x, t_i)$ first exceeds zero, where $t_0, t_1, \ldots, t_n$ are the time-points used by the chosen time-discretisation technique. Another measure we make use of is the conduction velocity (CV), defined as the speed of electrical propagation in a particular direction d with $\|d\| = 1$, where the electrical wave is considered to have reached a point x at time $A(x)$. Note that in order to measure this we need to choose two measurement points x and y such that $\frac{y-x}{\|y-x\|} = d$, and the wavefront propagation must be such that it is approximately locally perpendicular to d whilst it travels along the line joining x and y. We employ these as measures of the error in the one-, two- and three-dimensional cases, and note that while they have their imperfections, they are suitable for our purposes.

1.3 Thesis Structure

In Chapter 2, we review briefly the relevant physiology, and explain how computational models are used to describe it. Chapter 3 reviews the literature relating to monodomain simulation, discretisation techniques, preconditioning, and adaptivity. In Chapter 4 we explain the finite element method and introduce the hierarchical high-order form that we use. In Chapter 5 we show how to apply it to the monodomain system, including the correct treatment of the cardiac cell model, prove an *a priori* error estimate and show that it agrees with experimental simulation data. We also perform simulation on a variety of test domains in one, two and three dimensions in order to evaluate the accuracy and efficiency properties. In Chapter 6 we introduce our adaptive scheme, beginning with the *a posteriori* error indicator that we use to drive the adaptivity. We go on to present

our adaptive strategy, and show in one spatial dimension that the error control is very effective when it comes to holding the global error at a particular target level. We then evaluate the efficiency using similar test-problems to those in Chapter 5, demonstrating considerable improvements. We show the adaptive method in use on a two-dimensional re-entrant case, a standard three-dimensional cardiac test-problem and on a realistic rabbit heart geometry. Full evaluation of the results is given in Chapter 7, and some potential future directions are discussed in Chapter 8.

Chapter 2

Background

This chapter explains the multiscale biology that is modelled mathematically by cardiac cell ODE systems and the monodomain equations. It includes the heart as a whole, the cellular structure of the myocardium, and the ionic currents which cross the membranes of the individual cardiac myocytes during an action potential.

2.1 The Heart

Fundamentally, the heart is a mechanical pump with the essential dual tasks of keeping nutrients flowing to every cell in the body and pumping waste carbon dioxide to the lungs for excretion. The anatomy of the human heart is given in Figure 2.1; as for all mammals the structure is fully-divided into left and right sides which power systemic and pulmonary circuit flow respectively. Each side is split into two chambers, the smaller atria and the much larger and more powerful ventricles.

Upon completing a circuit of the body, deoxygenated blood arrives in the right atrium via the vena cavae. From here it is pumped via the tricuspid valve into the right ventricle. As the ventricles contract, the blood is then ejected from the right ventricle and driven around the pulmonary circuit where it becomes oxygenated in the lungs before returning

to the heart, arriving this time at the left atrium. From there, the blood then passes through the mitral valve into the left ventricle, the most muscular of the four chambers. The blood is ejected from here under high pressure to circulate through the rest of the body before returning once again to the right atrium.

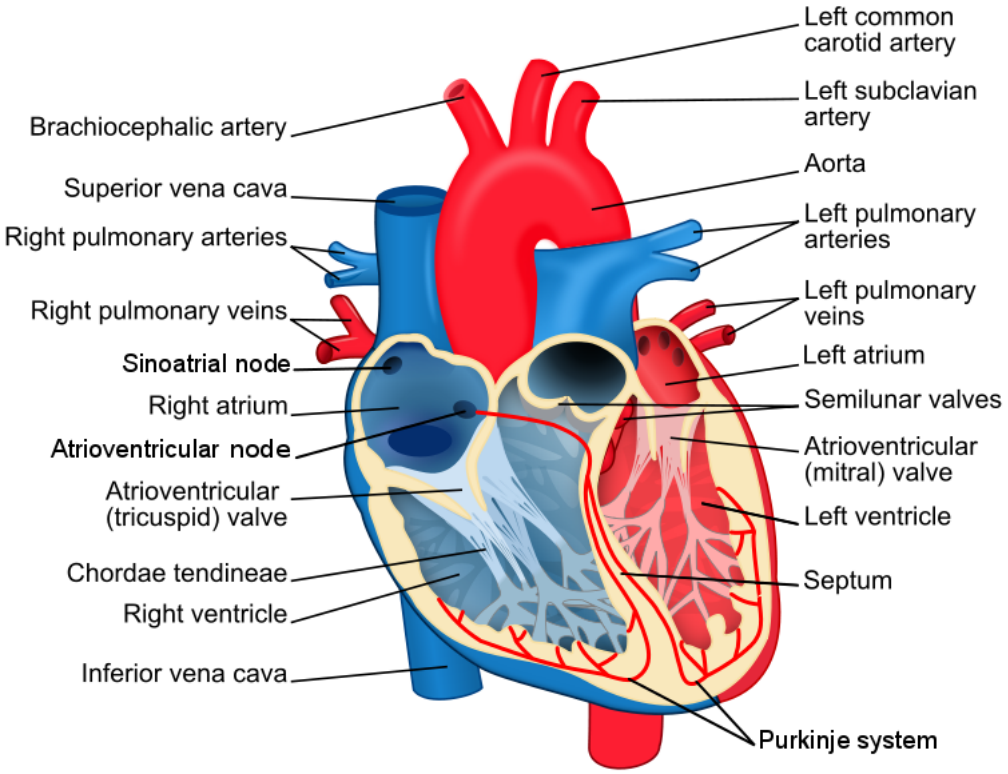


Figure 2.1: The human heart. Modified version of original image [30], used under the Creative Commons Attribution-Share-Alike 3.0 Unported license.

2.1.1 The Structure of the Myocardium and the Behaviour of Cardiac Myocytes

Making up the majority of the tissue in the heart walls is the myocardium, a layer of cardiac muscle tissue which lies between the endocardial lining on the interior of the chambers and the epicardium which is the exterior layer of the heart. The myocardium consists of roughly cylindrical, electrically excitable, contractile cardiac myocytes which are approximately 100 μm long and 15 μm in diameter. At their ends, the myocytes

are connected to their neighbours by gap junctions; these are pores which connect the cytoplasm between neighbouring cells and have a diameter of approximately 20 Å. Important for our interest in electrophysiology, the gap junctions allow for the flow of ions between the cells, resulting in an electrically-connected syncytium known as the intracellular space. The cells sit within the extracellular space, and are arranged into thin fibres, aligned axially; see Figure 2.2. These fibres are in turn arranged into sheets of myocardial tissue [6, 31].

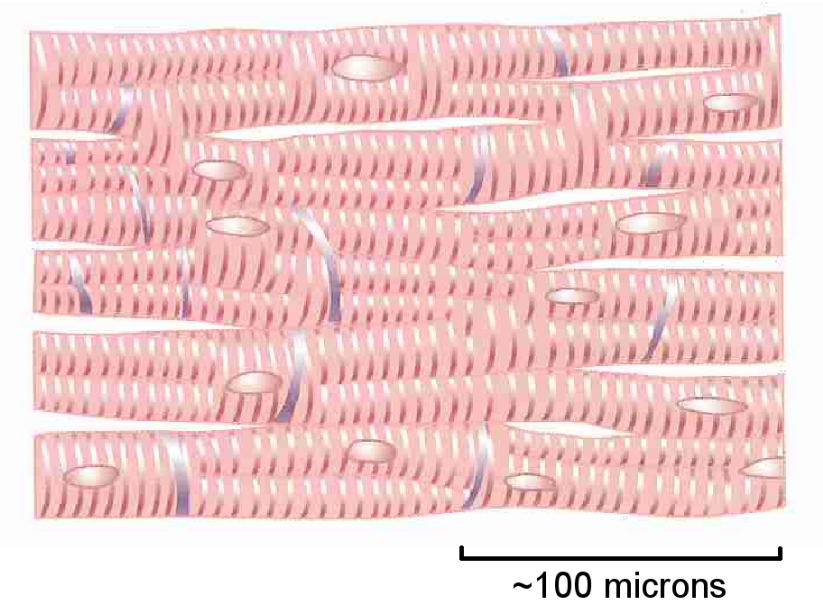


Figure 2.2: The arrangement of cardiac myocytes into fibres by connections at the gap junctions, creating a syncytium. Scan of Figure 9-2 from [31], modified to include an approximate scale.

Cardiac myocytes at rest maintain a transmembrane potential difference of approximately -80 mV, with the potential in the intracellular space more negative than in the extracellular. Upon being excited by an electrical stimulus, the cell membrane depolarises and then the difference becomes positive, reaching a peak at around 40 mV. This is caused by the flux of ions across the cell membrane, driven by ionic gradients and ion exchangers embedded in the cell membrane. Which particular ionic species is flowing varies throughout the cycle, see Figure 2.3. Broadly, depolarisation is caused by the fast inward sodium current, the plateau phase in the midsection of the action potential is

the result of the balance between the inward L-type calcium current and an outward slow delayed rectifier potassium current, and repolarisation is due to an outward flux of potassium ions in the inwardly rectifying current. These are the most important currents, each flowing through ion channels which open at specific points in the action potential as appropriate. There are also various ion exchangers and pumps involved, some of which are shown in Figure 2.3 [32]. Cardiac cell models attempt to capture the behaviour of these ionic currents.

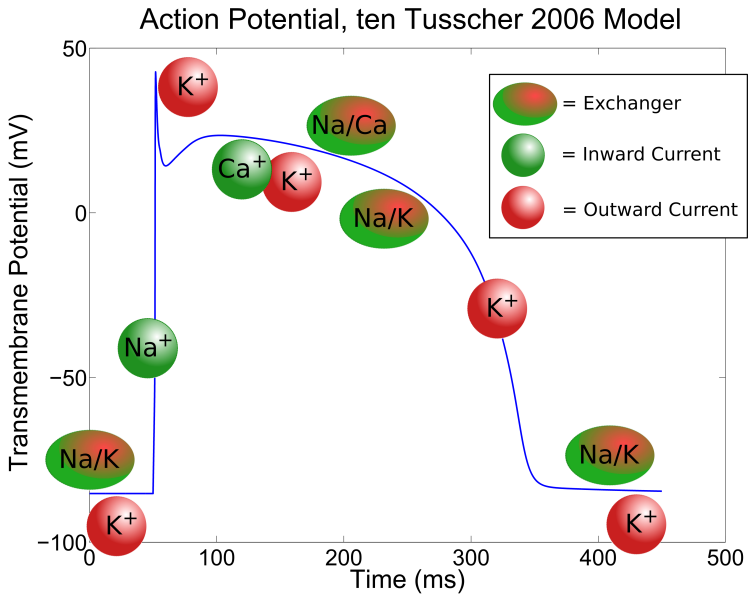


Figure 2.3: Schematic representation of the ionic flux across the cardiac cell membrane during an action potential.

2.1.2 The Heart Beat

The coordinated contraction of the myocytes which makes the heart beat is orchestrated by spatial waves in the electric potential difference between the intracellular and extracellular spaces, which we refer to as the transmembrane potential. These waves are caused by triggered pulses in the transmembrane potential on a cellular level, called action potentials (APs), triggering further APs in neighbouring cells via tissue-level electrical conductivity. Under physiological conditions, APs are initiated by the pacemaker cells

in the sinoatrial node, which is located in the right atrium, see Figure 2.1. Waves of electrical activation travel across the atria, causing contraction. The electrical impulse travels through the atrioventricular node and through the ventricular septum through the Purkinje specialised rapid conduction system (Figure 2.1), providing very fast distribution of the electrical impulse. The Purkinje system forms electrical connections with the ventricular myocytes at purkinje-muscle junctions, from where the action potential wave spreads out through the ventricular myocardium, which contracts in response [33].

2.1.3 Models of Cells and Tissue

For isolated cells, APs can be simulated using a system of ODEs [34, 35, 36, 37], or [38] for a detailed exposition, describing how the cellular machinery of the myocytes controls the flux of ionic species through ion channels across the cell membrane; we expand on this in Section 2.2. Full cardiac tissue simulation is made tractable via a homogenisation procedure which does away with the individual cells, modelling the tissue instead as two compartments representing the intracellular and extracellular spaces; both are considered to exist at every point in the domain. The spaces are electrically-isolated except for the transmembrane ionic currents between them, which are given by PDE analogues of the isolated cell ODE models. This homogenisation results in the bidomain reaction-diffusion-type system of differential equations describing AP propagation [5, 39, 40]; with σ_i, σ_e the intracellular and extracellular conductivity tensors respectively, ϕ_i, ϕ_e the intra- and extracellular potentials, $I_{stim,i}$, $I_{stim,e}$ the intra- and extracellular stimulus currents, $u = \phi_i - \phi_e$, and with all other symbols as for System (1.1), these form the problem *for some* $T > 0$ *and for all* $t \in [0, T]$, *find* $\phi_i(x,t)$, $\phi_e(x,t)$ *and* $w(x,t) = (w_1(x,t), \ldots, w_m(x,t))^T$

such that

$$C_m \frac{\partial u}{\partial t} - \frac{1}{\beta} \nabla \cdot (\sigma_i \nabla \phi_i) - I_{ionic}(u, w) = I_{stim,i}(x, t) \text{ in } \Omega, \tag{2.1a}$$

$$C_m \frac{\partial u}{\partial t} + \frac{1}{\beta} \nabla \cdot (\sigma_e \nabla \phi_e) - I_{ionic}(u, w) = -I_{stim,e}(x, t) \text{ in } \Omega, \tag{2.1b}$$

$$\frac{\partial w}{\partial t} - g(u, w) = 0 \text{ in } \Omega, \tag{2.1c}$$

$$\hat{n} \cdot (\sigma_i \nabla (u + \phi_e)) = 0 \text{ on } \partial\Omega, \tag{2.1d}$$

$$\hat{n} \cdot (\sigma_e \nabla (\phi_e)) = 0 \text{ on } \partial\Omega, \tag{2.1e}$$

$$\phi_i(x, 0) = \phi_i^0(x) \ \forall x \in \Omega, \tag{2.1f}$$

$$\phi_e(x, 0) = \phi_e^0(x) \ \forall x \in \Omega, \tag{2.1g}$$

$$w(x, 0) = w^0(x) \ \forall x \in \Omega. \tag{2.1h}$$

This problem consists of two PDEs describing the electrical conduction in the intracellular and extracellular spaces coupled to the PDE system for the transmembrane ionic currents and concentrations w. As we shall demonstrate in this work, we must be careful how we discretise the latter. The two spaces display anisotropic conductivity due to the sheets and fibres; in order to capture this, each has an associated conductivity tensor. Approximating these as being proportional to one another, this system can be reduced to the monodomain: a single PDE describing the dynamics of the transmembrane potential together with a formulation for w that remains unchanged from the bidomain, with conductivities in σ given as the harmonic mean of the intracellular and extracellular conductivity values from the bidomain. For more detail, see for example [6, 41].

The results obtained with the monodomain equation will not be identical to those found with the bidomain, differing in conduction velocity (CV) by around 2% [42], but because of the difference in computational effort required to solve the two forms, researchers use the monodomain where possible. Monodomain simulations can reproduce most of the behaviour seen when using the bidomain, including some which involve phenomena which once required the bidomain such as bath-loading effects [43], but excluding

defibrillation due to the need to simulate virtual electrodes [44]. The techniques that we present here are expected to extend without difficulty to the bidomain.

We work here with the monodomain system of equations with a cell model, more details about which can be found in [6].

2.2 Computational Modelling of Electrophysiology

2.2.1 The Model of Hodgkin and Huxley

The foundations for the computational modelling of cardiac electrophysiology were laid by Hodgkin and Huxley in their 1952 paper detailing a system of ODEs describing how the action potential in a squid giant axon is formed by the flux of ions across the cell membrane [45]. This early success earned them the 1963 Nobel Prize in Physiology or Medicine, and was possible due to the large size of this type of axon making experimentation possible with the techniques of the time.

They formulated their equations to describe the flux of sodium I_{Na} and potassium I_K, and other ions such as chloride (as a lumped "leakage current", I_l) across the nerve cell membrane as

$$I_{total} = I_{Na} + I_K + I_l,$$

where each of the individual currents I_X is given in terms of an ionic conductance g_X, the transmembrane potential difference u and a reversal potential u_X as

$$\begin{aligned} I_{Na} &= g_{Na}(u - u_{Na}), \\ I_K &= g_K(u - u_K), \\ I_l &= g_l(u - u_l). \end{aligned}$$

The conductances in the above are computed using

$$\begin{aligned} g_{Na} &= m^3 h \bar{g}_{Na}, \\ g_K &= n^4 \bar{g}_K, \end{aligned}$$

where the $\bar{g}$ terms are the maximum conductances to the subscripted ions of the cell membrane, and h, m and $n \in [0, 1]$ are called *gating variables*, and are related to the proportion of ion channels within the membrane which are in the open state, allowing ions to flow. These variables are evolved using the ODEs

$$\frac{dn}{dt} = \alpha_n(u)(1-n) - \beta_n(u)n, \tag{2.2a}$$

$$\frac{dm}{dt} = \alpha_m(u)(1-m) - \beta_m(u)m, \tag{2.2b}$$

$$\frac{dh}{dt} = \alpha_h(u)(1-h) - \beta_h(u)h, \tag{2.2c}$$

which are known as *Hodgkin-Huxley form gating variables*, and the α_X and β_X are experimentally-derived functions which depend only on the transmembrane potential u. The action potential time course can be computed using these equations together with

$$\frac{du}{dt} = -\frac{1}{C_m} I_{total},$$

where C_m is the constant capacitance of the cell membrane per unit area. The solution to this system of equations yields the action potential for a single nerve cell under the assumption that the potential difference is the same across the whole of the cell membrane.

For the case where this assumption is not made, Hodgkin and Huxley noted that it was possible to write down a PDE to describe propagation along a nerve axon as

$$\sigma \frac{\partial^2 u}{\partial x^2} = C_m \frac{\partial u}{\partial t} + I_{ion}, \tag{2.3}$$

where they modelled the axon as one-dimensional in the spatial variable x, and where σ

is the conductivity of the membrane. However, they felt that it was "not practicable" to solve this equation, presumably due to the available computing power given that it took Huxley around three weeks to compute an action potential under the uniform potential difference assumption using a mechanical calculator [46].

Note that despite the formal similarity between Equations (2.3) and (1.1), they differ in that they attempt to model different things. Equation (2.3) models propagation along the membrane of a single cell, whereas Equation (1.1) derives from a homogenisation process for propagation in tissue consisting of many millions of cells.

2.2.2 Cardiac Cell Models

Many cardiac cell electrophysiology models are designed using the principles introduced by Hodgkin and Huxley for the squid giant axon. There are far too many of these in the literature to cover in this thesis, so we examine just a few here. For a more complete exposition, see for example [38, 47, 48].

The first cardiac cell model was published by Noble in 1962, describing the action potential in a cardiac Purkinje cell [49]. This built directly on the work of Hodgkin and Huxley, modifying their model to make it suitable for the cardiac myocyte by adding a second potassium current and adjusting the constants in the model appropriately, producing a model which "correspond[ed] quite well with the observed behaviour of Purkinje fibres". The design of this model required that the sodium current be active over an unexpectedly large range of membrane potentials, which hinted at the contribution of as-yet unknown currents to the cardiac action potential [47]. Like many of the models that were to come later, Noble's model utilised Hodgkin-Huxley form gating variables; a schematic of the modelled ion channels is shown in Figure 2.4.

Modelling ionic concentrations in addition to gating variables and the transmembrane potential, the first well-used [48] ventricular myocyte model was introduced by Beeler and

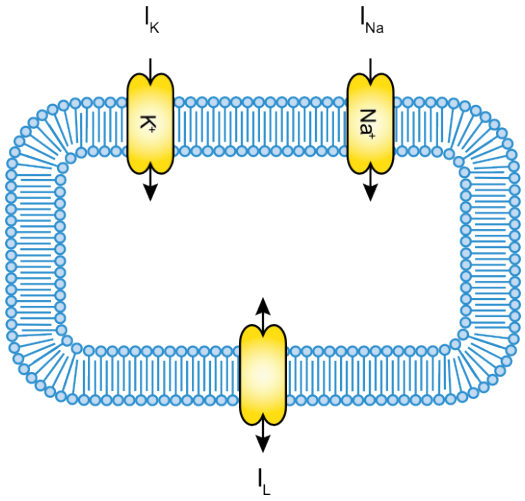


Figure 2.4: Schematic of the Noble 1962 cell model [49] showing specific directional and non-specific non-directional ion channels. Clockwise from the top-left, the currents are potassium, sodium, and a leak current which was attributed at least partially to chloride ions. Used under the Creative Commons Attribution 3.0 License; attribution: [50]

Reuter in 1977 [51]. This model incorporated a differential equation

$$\frac{d[Ca^{2+}]_i}{dt} = -10^{-7} I_s + 0.07\left(10^{-7} - [Ca^{2+}]_i\right),$$

to describe the intracellular calcium concentration $[Ca^{2+}]_i$ in terms of the slow inward calcium current I_s. Note that this is, by necessity, not of Hodgkin-Huxley form; this has implications for simulation as it means we cannot immediately take advantage of numerical integration methods such as that of Rush-Larsen [52]. The schematic of this cell model is shown in Figure 2.5; it was an important step forward is it sought to investigate the salient differences between Purkinje and ventricular myocytes.

The first complete guinea-pig ventricular model is that of Luo and Rudy 1991 (LR1) using eight ODEs (six are Hodgkin-Huxley gating equations, one models the transmembrane potential difference, and one the intracellular calcium concentration) [34]. It was built as a reformulation of the Beeler-Reuter model, using data from single-cell and single-channel measurements in its description of the ionic currents. It successfully reproduces physiological phenomena such as the maximum upstroke velocity and APD alternans [54].

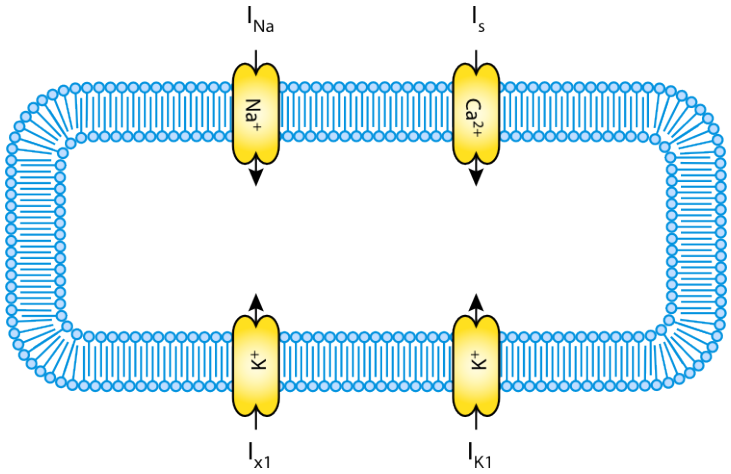


Figure 2.5: Schematic of the Beeler-Reuter 1977 cell model [51] showing specific directional ion channels. Clockwise from the top-left, the currents are the fast sodium, the slow inward calcium, the time-dependent potassium current and the voltage- and time-independent potassium current. Used under the Creative Commons Attribution 3.0 License; attribution: [53]

From a computational point of view, it contains some unnecessary discontinuities which must be treated before we make use of the model in this thesis; this will be discussed further in Section 5.2.3. A schematic for this cell model is given in Figure 2.6.

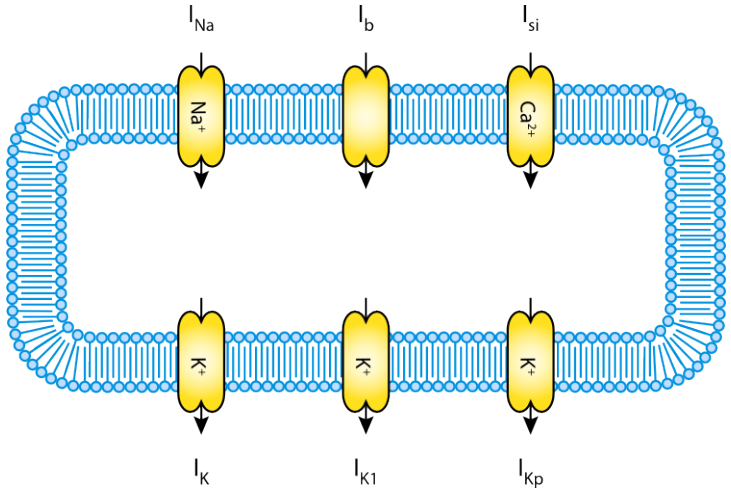


Figure 2.6: Schematic of the Luo-Rudy 1991 cell model [34] showing specific and non-specific directional ion channels. Clockwise from the top left, the currents are fast sodium, background, slow inward, plateau potassium, time-independent potassium and time-dependent potassium. Used under the Creative Commons Attribution 3.0 License; attribution: [55]

The first standard model to use human ventricular data was that of ten Tusscher et al. in 2004 [56], and has versions for epicardial, mid-myocardial and endocardial cells. In this work, we use its minor extension [57] which improves upon the original model's handling of calcium to include concentrations in the diadic subspace, cytoplasm and sarcoplasmic reticulum (see Figure 2.7), and consists of 17 ODEs, twelve of which are Hodgkin-Huxley gating variables. This is the second cell model which we make use of in this work, as it was prescribed for the evaluation of 3D simulation by Niederer et al. [58], whose work we make use of in Section 5.5.2.

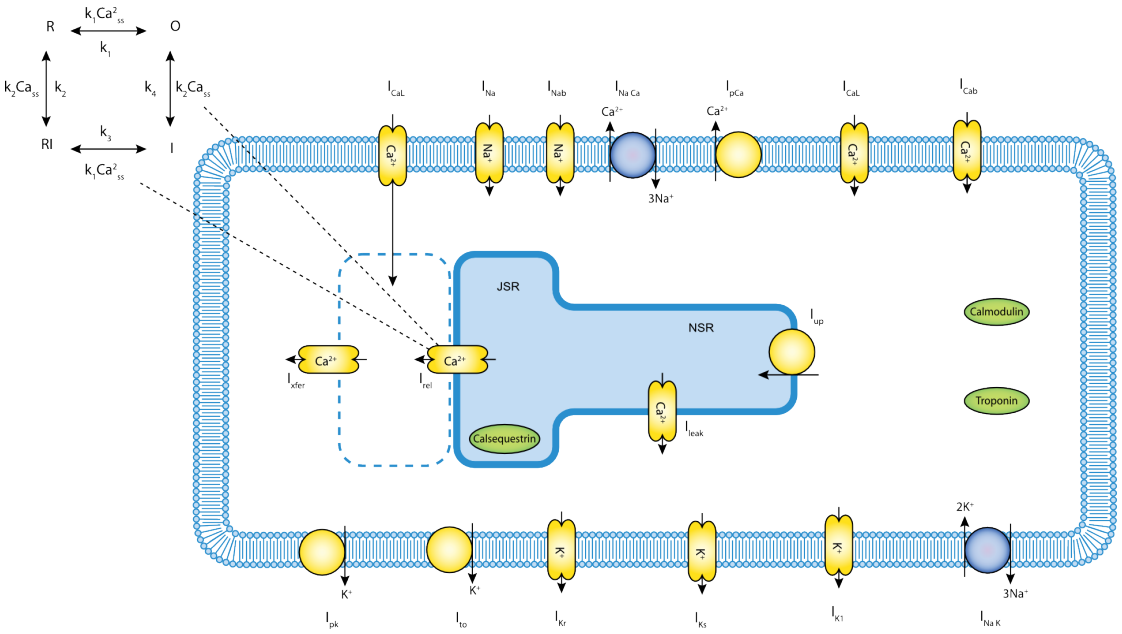


Figure 2.7: Schematic of the ten Tusscher 2006 cell model [57] showing directional ion channels, ion pumps and exchangers and calcium concentrations in the sarcoplasmic reticulum, diadic space and cytoplasm. Clockwise from the top-left, the membrane currents are L-type calcium (subspace), fast sodium, background sodium, sodium-calcium exchange, plateau calcium, L-type calcium, background calcium, sodium-potassium pump, inward rectifier potassium, slow delayed rectifier potassium, rapid delayed rectifier potassium, transient outward potassium and plateau potassium. Used under the Creative Commons Attribution 3.0 License; attribution: [59]

As we can easily see from Figures 2.4, 2.5, 2.6 and 2.7, there has been a substantial increase over the years in the amount of detail that cardiac cell models incorporate. While this does translate into increased computational cost, because of its essentially

decoupled nature the parallelisation of the cell model in tissue is trivial. This means that we will obtain almost perfect parallel compute-time scaling for the tissue-incorporated cell model as we increase the number of processors, so the demands of the cell model can be considered to be secondary to those of the tissue-level equations.

Chapter 3

Literature Review

This chapter covers the literature relating to monodomain discretisation techniques, accuracy, adaptivity and the difficulties associated with using the h-version FEM, investigations into preconditioning strategies, and some applications of cardiac simulation.

There is a very large body of work on simulating cardiac electrical propagation, most of which concentrates on the mono- or bidomain equations [5, 6]. These can be divided into two major groups: those which are concerned with using simulation as a tool for answering research questions [60, 61, 62, 63, 64, 65, 66], and those which are interested in developing the simulation techniques themselves in order to enable the other camp to perform faster, more complex and more accurate simulation studies and investigations [67, 68, 58, 13, 14, 43, 12, 16, 69]. This work falls into the latter category, but attempts to keep both in mind in order to avoid an unfortunate disconnect between the two which, when it occurs, is always detrimental to the field as a whole.

Probably the most frequently-seen (and most worrying) manifestation of this disconnect is on the simulation-study side, where researchers without a good understanding of numerical methods have applied tools in a poor or incorrect manner; this sort of problem typically manifests itself as under-refined spatial or temporal resolutions coupled with a lack of convergence-testing, which can result in conclusions being drawn from highly

inaccurate data. For example the work presented in [70] involves the following rather odd approach to convergence. The authors choose conductivity parameters that reproduce a desirable biological result (a specific QRS duration, which is the time taken for the depolarisation of the ventricles and is measured by the time between two points on the clinically-important electrocardiogram; for historical reasons these points are known as Q and S), then test whether or not their results were converged by refining the mesh. Upon doing this, they observe a 16% drop in the QRS duration, suggesting a lack of convergence, but then go on to claim that

> "[S]ince the conductivity parameters had originally been chosen to match the observed QRS duration [in patient data], when we rescale the conductivities in the refined model to match the QRS duration, the RMS difference between activation times in the coarse and fine models was 1.2%. Hence for the current model, the activation time solutions were converged satisfactorily."

This is an example of the sort of trap that excellent researchers can easily fall into if they do not have a good understanding of both the numerical and biological aspects. Similar examples occur in Talbot et al. [71], and in a perhaps less concerning form in [72]. A far more common mistake is for workers to use meshes which are too coarse without saying anything at all about convergence. For example, [73] performs electrical simulation on three different canine heart meshes, having 623, 3,736 and 9,623 nodes. This is clearly far too coarse for a linear approximation to obtain anything like converged results. Of course, this is not always the case; there are plenty of careful workers who sensibly adjust the space- and time-step size depending on the problem at hand (conductivity, cell model choice, etc.) [74], but it is a shame to have high-quality work marred by avoidable misunderstandings.

Conversely, a knowledge of the mathematics without an understanding of the relevant biology can be equally harmful; in this present work, we would not be able to perform some of the required numerical analysis if we were to blindly follow the behaviour of the equations. Instead, we will demonstrate how an awareness of the fact that the equations have been designed to represent the behaviour of a physical system can be useful in

their analysis, allowing us to argue for desirable mathematical properties that we cannot prove. For example in Chapter 5 we will require a Lipschitz continuity condition that we are unable to prove, but we can argue for using biology as follows. *We do not know whether our particular equations are Lipschitz, but if they are, then the analysis we wish to perform is possible, and if they are not then the equations ought to be replaced with Lipschitz versions because the biological reality demands it.* We will build on this argument in Section 7.1.1.

In light of this, the following literature review attempts to cover the relevant aspects of numerical analysis and computational physiology, as without a good understanding of both we cannot hope to produce high-quality science.

3.1 Numerical Literature

3.1.1 Discretisation Techniques

The method that we use for spatially discretising our equations in this work is the finite element method (FEM) for obtaining approximations to the solution of partial differential equations. It was introduced in its present form in 1942 by Richard L. Courant as an appendix to his paper on variational methods [75], although it does not then appear in the literature again for another sixteen years [76]. It was further developed by engineers during the late 1950s and early 1960s as a method of solving equations in structural engineering [77], and has since developed into a general tool for the numerical approximation of solutions to PDEs.

The common forms of the FEM are the h-version, in which convergence is achieved by discretising the domain into successively smaller (and thus a larger number of) elements, the p-version in which convergence is achieved by increasing the interpolation order either globally or locally [19], and the hp-version which takes a combination of both approaches [78]. These methods are not necessarily used adaptively during the simulation; prior knowledge about the requirements for sufficient convergence of the system can be used

to select the appropriate values of h and p in advance. Generally this is the approach taken for state-of-the-art cardiac electrophysiology simulation, with p=1 and h in the approximate range 50-500 μm. Even towards the finer end of this range, potentially concerning errors can exist in numerical solutions, particularly when working with low electrical conductivity. This is one reason why accurate solutions are difficult to obtain in a reasonable period of time. We shall see more of this when comparing the accuracy of the state-of-the-art to that of our approach.

A recent paper by Niederer et al. [58] compared eleven cardiac monodomain codes using FEM or the finite difference method (FDM) in terms of the accuracy of their activation times at various points in a simulation domain. This demonstrated the wide range of solution variability between codes solving identical problems, showing that care that must be taken to ensure that no accidental error is introduced by poor choice of solution technique. There is not much agreement between the different codes, and this paper lacked a very-high resolution reference solution, so while this is a step in the right direction, the evaluation that was possible was limited.

The powerful ability of FEM to capture complex geometries such as the complete myocardium, together with its very simple implementation requirements for zero-flux boundary conditions makes it the most widely used discretisation technique in the field. Those codes tested in Niederer et al. [58] which are capable of simulating general geometries are all based on FEM; the FDM implementations therein are restricted to regular grids. Additionally for our purposes, FEM's use of variational forms of the system equations affords us a useful framework in which to perform convergence analysis. Aside from FEM and FDM [79, 80], other discretisation methods include finite volume [81, 82, 10] and mortar element [18] methods, although these are far less common.

There are two major components to the time discretisation of the monodomain system. These correspond to the time derivatives which occur in the tissue level Equation (1.1a) and those which occur together in the vectorised system for the cell model, Equation (1.1b). Explicit forward Euler discretisations can be used for both of these components

due to their simplicity, avoiding the need for inner iterations that implicit methods would require. At the tissue level, in practice it is as simple to use a semi-implicit method, treating implicitly any terms which can be algebraically manipulated to avoid the inner iterations, and this results in better stability properties. A range of methods for the tissue time-integration are studied in [69]; here the conclusion is reached that semi-implicit higher-order methods in time are optimal for efficiency amongst the ten methods studied. For the cell model, a wide range of time-stepping methods have been investigated. Beyond the standard implicit and explicit Euler approaches, a widely-used method for time-marching Hodgkin-Huxley form gating variables (see the equations of System (2.2)) is the Rush-Larsen method [52], which is popular because of its excellent efficiency properties coupled with ease of implementation, although it is not applicable to the non-gating state variables in the cell model [83]. Other numerical methods building on this approach have been investigated, such as the Generalised Rush-Larsen method which provides a way of dealing with the non-gating variables by also treating them using exponential integrators [84]. These methods have been shown to outperform basic Rush-Larsen when stiffness occurs in the non-gating variables [83].

Also based on Rush-Larsen, time-adaptive schemes have been developed [85]; this method uses an error indicator to robustly control the time-step for the cell model. It remains to be seen whether it generally outperforms non-adaptive methods, and it is not clear how best to implement this sort of scheme for the cell model when it is embedded in tissue. Other approaches taken to temporal adaptivity include those based on thresholding of state variable derivatives [52, 22] or on the ionic currents [15], meaning that neither of these use estimates of the error to control the time-step.

There is some existing work on nonadaptive higher-order finite element methods to-date. For example, Maggio et al. [28] describes the use of globally-fixed-order elements on hexahedral meshes (and implies that such a mesh is necessary for this method, but of course there is no reason why we could not define a polynomial basis on a tetrahedron; this is a claim described as "the main drawback" of this type of method in Bordas et al.

[26]). They claim that one of the advantages of their implementation is that the mass matrix is diagonal always, meaning that there is no need to consider mass-lumping; a closer examination of their technique [29] reveals that this is only the case because the numerical integration rule they have used integrates polynomials up to degree $N-1$, where the polynomials being integrated are of degree N; this is similar to mass-lumping. They note that different preconditioners will be required for p-version problems due to the different structure when compared to linear elements. They suggest that nonadaptive p-version will be able to provide between 2 and 3 times speed-up, at the expense of roughly doubling the the memory requirements. However, their studies compare the use of tetrahedral meshes with the linear-basis elements to hexahedral meshes with the p-version; it is not clear that using hexahedra is the best approach to p-version FEM due to the limited geometric flexibility, so it is important to investigate the p-version on tetrahedral meshes. The improved accuracy available from high order elements is demonstrated by Maggio et al. [27] using test problems with no cell model (although they do mention the Luo-Rudy 1991 model [34], they never use it). Their later work [86] indicates that a speed-up of roughly one hundred times over linear FEM is possible in the test-case where no cell model is included in the simulations, and that this reduces to more like twenty times faster when the cell model is added. None of this p-version work by these authors has involved using local adaptivity in space, as this thesis does, and given that the quoted twenty-times speed-up is non-adaptive, dynamically selecting linear basis functions after the initial upstroke and retaining them for the majority of the action potential, as controlled by a local error indicator, should be expected to allow for an even greater improvement in simulation speed.

3.1.2 Adaptive Discretisations in Space: Finite Elements

Spatially adaptive methods tried to-date in the literature have met with limited success when applied to the cardiac monodomain and bidomain equations. Bernabeu et al. [87] note that there is only a "limited selection" of papers on adaptive techniques for the

bidomain equations, despite the popularity of such techniques in numerical analysis in general. They list the following four reasons for this:

1. Significant computational effort is required to compute the mesh required for each time-step

2. When the mesh is altered the [system matrices] must be re-computed, and this itself requires a large volume of computations

3. Each node introduced at refinement requires the generation of state from a cell model

4. In modelling fibrillation, the refinement is required to change rapidly.

The majority of the work that has been done on spatial adaptive methods for cardiac simulation has been based on h-refinement [40], which we do not consider here in this work. The reductions in computational time that have been observed using these methods range from 10% to two orders of magnitude faster [26]. However, with the exception of Southern et al. [88], we are not aware of any which have been shown to provide an increase in efficiency in 3D [87]; this is likely due to the reasons enumerated above.

Methods for adaptivity in the literature include that of Whiteley [15], which uses the biology to drive h-adaptivity by identifying steep gradients in the cell state variables and adapting the local mesh element size based upon that. This approach has the advantage of using a refinement criterion which is very cheap to compute, but the error is controlled in a heuristic rather than analytic manner so we can not necessarily have confidence in the accuracy of the results, and it does not circumvent the problems listed above. The same can be said of the paper's approach to temporal adaptivity: it is a scheme based on identifying when the ionic currents have dropped below some threshold value everywhere in the domain, and then switching the time-step to a larger size until the ionic currents recross the threshold. Having said this, with the loosest thresholding for adaptivity, the presented results in [15] show an order of magnitude of speedup with space

and time adaptivity in two-dimensional domains. Regarding the high computational cost of recomputing the system matrices in h-adaptive approaches, the paper also notes that when around half of a 1 cm $\times$ 1 cm simulation domain is refined from 1 mm spacing to 0.1 mm spacing, the additional cost of rebuilding the matrices balances the speedup gained from the system being smaller than if the fine spacing was used nonadaptively everywhere. It seems likely that working with h-adaptivity in three-dimensions would result in an even more time-consuming matrix reassembly process.

Colli-Franzone et al. [17] use the adaptive library KARDOS [89] to implement h-adaptivity using a hierarchical error estimator, together with a method of adaptively choosing the time-step based on a temporal error estimator. Using this method, two-hundred times fewer finite difference nodes than a non-adaptive method would need are used to produce the same L^2 error in the solution. Pennacchio [90] takes a mesh refinement approach with the mortar element method, decomposing the domain into subdomains, identifying which subdomains contain wavefronts and selecting fine grids to use for them, whilst choosing coarse grids for those without wavefronts. The method is shown to provide solutions of equal accuracy to a standard linear finite element solution in around a quarter of the time. Again using the KARDOS library, Deuflhard et al. [91] presented simulations performed on a realistic cardiac geometry in a state of fibrillation, with 800 ms requiring six weeks of CPU time on a 16 core shared memory machine, despite this adaptivity. This result lead the authors to conclude that improvements to the KARDOS system were essential.

One way of mitigating the cost of h-adaptivity is to perform the mesh adaptation and matrix reconstruction every few time-steps rather than on each step [92]. The advantage in terms of reducing the cost of adaptivity is obvious, but it requires additional heuristics such as the introduction of also-refined border-zones around the area marked for refinement to prevent the wavefront from leaving the refined region before the next adoption takes place [93].

The FEM adaptive p-version that we pursue in this work provides a potential solution to the computational overhead problems encountered by h-adaptivity; in the case of the p-version the system matrices are calculated once at the beginning of the simulation, and then spatial adaptivity is reduced to switching basis functions on or off by including or excluding whole rows and columns from the pre-computed (for efficiency) everywhere-maximal-degree system matrices, see Section 6.3. This, together with the fact that our actual mesh never changes means that the cell discretisation never changes, mitigates the effect of points 1-3 listed above. Additionally, the recent work of Southern et al. [88] demonstrates that considerable 3D whole-heart efficiency improvements are possible with an anisotropic h-adaptive approach with dynamic load balancing in parallel. Importantly, these workers show that h-adaptivity is beneficial despite the vast majority of the CPU time being devoted to refining the mesh. This is consistent with points 1-3 above, and indicates that avoiding this overhead using the p-version could provide great benefits.

3.1.3 Adaptive Discretisations in Space: Other Methods

Beyond the finite element method, other approaches to spatial adaptivity include the work of Cherry et al. [94, 16] who use adaptive mesh refinement driven by thresholding on the local truncation error, which is estimated using Richardson extrapolation [93]. The method uses a finite difference scheme, and they report a reduction in computational time and memory requirements by a factor of at least 30 for a single propagating wavefront in a 3D monodomain simulation using the Luo-Rudy 1991 cell model. Although this method is not closely related to the state-of-the-art finite element approaches, it hints at the benefit which is available to simulation from spatially adaptive techniques. This paper additionally points out that while adaptive methods could be of limited benefit once fibrillation is in progress, depending on how closely packed wavefronts in such a situation are, they can be extremely useful for investigations of the onset of fibrillation where the wavefronts are not so dense in the domain (for example, during pacing or tachycardia). Having said this, visual inspection of fibrillation simulations on realistic geometry [95]

suggests that the wavefronts are still sparse enough in the domain to allow for much speed-up from using the adaptive p-version. Both [15] and [16] discuss parallelisation but neither implement it.

3.1.4 Preconditioning

The discretisation of System (1.1) results in a system of linear equations which must be solved. Because this system is too large to handle using a direct method it must be solved iteratively, requiring us to use a *preconditioner* to reduce the number of iterations required. We expand on this brief explanation in Section 4.6, but note here that a preconditioner can be thought of as a method of generating an approximate and computationally-cheap inverse to the matrix of the linear system. Investigations of preconditioners for this problem in the literature include the work of Bernabeu and Kay [14], who compared block-diagonal (BD), lower-triangular diagonal upper-triangular (LDU) and algebraic multigrid (AMG) preconditioners, concluding that the BD and LDU gave the best performance at fine spatial discretisations, but noted that AMG can be the better choice when using large numbers of CPU cores or large values of h; this highlights the difficulty in choosing appropriate preconditioners. A review by Vigmond et al. [41] concluded that multigrid methods were in fact the fastest methods available, although they were comparing to incomplete LU (ILU) factorisation. Pennacchio and Simoncini showed that AMG iteration counts for the bidomain problem are essentially mesh-independent [96, 97, 98]. Dos Santos et al. [99] investigated additive Schwarz [100], multigrid and ILU preconditioners in a parallel implementation, concluding that multigrid is the most efficient, followed by additive Schwarz. Pavarino and Scacchi [101] looked at additive Schwarz preconditioners and concluded that their approach was "optimal and scalable" due to the fact that the iteration counts were unaffected by the number of processors they used. In later work [102], Scacchi finds that a particular Schwarz method gives the "best CPU times to date", although the comparison was only explicitly made with multigrid and Block Jacobi ILU.

As this cross-section of the literature indicates, choosing a preconditioner is not straightforward because the answer to the question "which is best?" is highly situation-dependent.

3.2 Research Applications

Monodomain and bidomain simulation has been used to investigate the role of drugs in arrhythmogenesis [10, 103, 104], patient-specific cardiac resynchronisation therapy [70], the effect of myocardial microstructure on propagation [95], the suggestion that a larger heart has increased APD dispersion increasing the risk of arrhythmia [105], the possibilities for training surgeons [71], and the mechanisms of defibrillation [106, 107]. This list is far from exhaustive, but the majority of these studies note concerns about computational efficiency, either in terms of the time required for simulation or the numerical accuracy that the study attained. Concerns about accuracy are not surprising, given that obtaining $< 1\%$ error in the conduction velocity requires mesh discretisations smaller than 50 μm along fibres and around 10 μm in the perpendicular direction [108], and given further that the required sizes will become smaller still when conduction velocity drops in the case of pathologies of interest such as ischemia [109].

Simulation studies are not limited to just the electrical behaviour; some workers study combined electrical and mechanical behaviour, in which the electrical propagation drives the contraction of the heart model, and often incorporates mechanoelectric feedback from the contractile model into the electrics. For example, this sort of simulation can be used to investigate strategies for, and predict outcomes of, cardiac resynchronisation therapy; this is a surgical procedure in which pathological asynchronous left- and right-ventricle electrical activation is repaired in order to improve cardiac output [110, 111]. The results generated by this sort of work are extremely promising, indicating that simulation is likely to have an impact on patient treatment decisions in the near future. However, such applications mean that erroneous or inaccurate simulation results could become

extremely dangerous, and so accuracy is important.

Generally in the field, numerical convergence results are not presented alongside simulation studies, and in a proportion of those that do provide such information, the methods used to assess convergence are poor. It should be the case that publications are required to present (or reference) proper accuracy analysis for each simulation study, so that readers may evaluate the presented results in the context of numerical accuracy.

Chapter 4

The Finite Element Method and the High-Order p-Version

In this chapter, we explain the finite element method by using several example PDEs of increasing complexity, building up towards the finite element discretisation of the cardiac monodomain equations, which we will see in Chapter 5. We introduce the commonly-used linear basis, and then the contrasting hierarchical basis for the finite element p-version that we use to produce computationally efficient discretisations. We also cover high-order nodal bases which are required to discretise the cardiac cell model spatially. We discuss briefly how these bases are constructed in practice using a reference element. Some possible preconditioners are introduced at the end of the chapter.

The finite element approach for the space discretisation differential equations is outlined in what follows. We work with the norms

$$\|\chi\|_{m,q} = \left(\sum_{0 \leq |\alpha| \leq m} \int_\Omega |D^\alpha \chi|^q \ dx \right)^{\frac{1}{q}},$$

with $\alpha = (\alpha_1, \ldots, \alpha_n)$, $|\alpha| = \sum_{i=1}^n \alpha_i$ and $D^\alpha := D_1^{\alpha_1} \ldots D_n^{\alpha_n}$, where $D_i := \frac{\partial}{\partial x_i}$ [112], and the Hilbert spaces

$$\mathcal{H}^m(\Omega) := \left\{ \chi \in L^2(\Omega) \mid \|\chi\|_{m,2} < \infty \right\}.$$

For convenience we shall often use the abbreviations $(a,b) := \int_\Omega a \cdot b \; dx^l$ and $\langle a,b\rangle_{\partial\Omega} := \int_{\partial\Omega} ab \; dx^{l-1}$, where we are thinking of $\partial\Omega$ as being part of the boundary of Ω. We use the Lebesgue measure dx so that we only need the functions to be defined almost everywhere.

4.1 The FEM for Laplace's Equation

4.1.1 Initial Discretisation of Laplace's Equation

It is instructive to begin at a simple level and proceed by incrementally adding to the complexity of the equations we are discretising when explaining the use of the FEM, so we begin by considering Laplace's equation: *find* $u(x)$ *such that*

$$-\Delta u = 0 \text{ in } \Omega, \tag{4.1a}$$

$$\hat{n} \cdot \nabla u = g_N \text{ on } \partial\Omega_N, \tag{4.1b}$$

$$u = g_D \text{ on } \partial\Omega_D, \tag{4.1c}$$

where Ω with polygonal boundary $\partial\Omega$ is an open subset of $\mathbb{R}^l$ for some l. The boundary consists of one portion $\partial\Omega_N$ on which we have Neumann boundary conditions g_N and another portion $\partial\Omega_D$ on which Dirichlet boundary conditions g_D are prescribed, and we have $\overline{\partial\Omega_N} \cup \overline{\partial\Omega_D} = \partial\Omega$, $\partial\Omega_N \cap \partial\Omega_D = \emptyset$. The outward-pointing unit normal $\hat{n}$ to $\partial\Omega$ is defined almost everywhere on the boundary. See Figure 4.1.

In the case where $g_N \equiv 0$, system (4.1) can describe the distribution of heat in an object with geometry described by Ω, where we have perfect thermal insulation on $\partial\Omega_N$ and fixed temperature distributions given by the function g_D defined on the boundary due to some part of the environment with fixed temperature contacting the object along $\partial\Omega_D$. Alternatively, slightly less intuitively but with more direct relevance to this work, it can describe the distribution of electrical potential in an object with applied voltages g_D on $\partial\Omega_D$ and perfect electrical insulation on $\partial\Omega_N$.

In order to solve this problem numerically, we must make it finite dimensional by

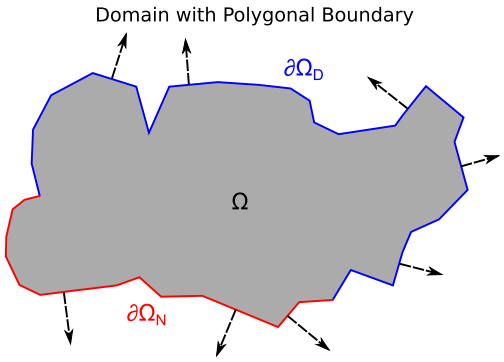


Figure 4.1: A two-dimensional domain with polygonal boundary, split into Neumann ($\partial\Omega_N$) and Dirichlet ($\partial\Omega_D$) components. The outward pointing normals $\hat{n}$ are shown at various points.

discretising it suitably. The procedure for doing so using the *Galerkin method* is as follows. We begin by re-writing Equation (4.1a) in the weak form; we seek some $u \in \mathcal{H}^1_D(\Omega) \cap \mathcal{H}^2(\Omega)$, with $\mathcal{H}^1_D(\Omega) := \{v \in \mathcal{H}^1(\Omega) \mid v = g_D \text{ on } \partial\Omega_D\}$, which satisfies

$$(-\Delta u, \chi) = 0 \quad \forall \chi \in \mathcal{H}^1_0(\Omega), \tag{4.2}$$

where $\mathcal{H}^1_0(\Omega) = \{v \in \mathcal{H}^1(\Omega) \mid v = 0 \text{ on } \partial\Omega_D\}$.

To deal with the boundary condition, we choose some fixed function $u_D \in \mathcal{H}^1_D(\Omega) \cap \mathcal{H}^2(\Omega)$, and write $u = u_D + u_N$ for the u_N that we are to determine. Thus, Problem (4.2) becomes *find* $u_N \in \mathcal{H}^1_0(\Omega) \cap \mathcal{H}^2(\Omega)$ *such that*

$$(-\Delta u_N, \chi) = (\Delta u_D, \chi) \quad \forall \chi \in \mathcal{H}^1_0(\Omega),$$

to which we apply Green's identity to obtain

$$\begin{aligned} (\nabla u_N, \nabla \chi) \; &- \; \langle \hat{n} \cdot \nabla u_N, \chi \rangle_{\partial\Omega_D} - \langle \hat{n} \cdot \nabla u_N, \chi \rangle_{\partial\Omega_N} \\ &= \; \langle \hat{n} \cdot \nabla u_D, \chi \rangle_{\partial\Omega_D} + \langle \hat{n} \cdot \nabla u_D, \chi \rangle_{\partial\Omega_N} - (\nabla u_D, \nabla \chi) \quad \forall \chi \in \mathcal{H}^1_0(\Omega). \end{aligned}$$

The use of the space $\mathcal{H}^1_0$ allows us to discard the integrals on $\partial\Omega_D$ as these are always

equal to zero, giving

$$(\nabla u_N, \nabla\chi) - \langle \hat{n}\cdot\nabla u_N, \chi\rangle_{\partial\Omega_N} = \langle \hat{n}\cdot\nabla u_D, \chi\rangle_{\partial\Omega_N} - (\nabla u_D, \nabla\chi) \qquad \forall\chi \in \mathcal{H}^1_0(\Omega),$$

which can be rearranged, and the problem re-stated (dropping the $\mathcal{H}^2$ requirement) as *find* $u \in \mathcal{H}^1_D(\Omega)$ *such that*

$$(\nabla u_N, \nabla\chi) = \langle g_N, \chi\rangle_{\partial\Omega_N} - (\nabla u_D, \nabla\chi) \qquad \forall\chi \in \mathcal{H}^1_0(\Omega); \tag{4.3}$$

the change of space is possible because we no longer have any second-order differential operators, and this is now the final continuous weak form of our problem. $u = u_N + u_D$ satisfying Equation (4.3) will satisfy Equation (4.1), assuming that it is sufficiently smooth for the second derivatives in Equation (4.1a) to exist.

The fact that we require Equation (4.3) to be satisfied for all such χ means that it really represents an uncountable infinity of conditions which must be checked before we know that we have found the solution u_N that we seek. Clearly this is unsuitable for translation into computer software, so we discretise by instead working in a finite dimensional subspace of $\mathcal{H}^1_0(\Omega)$, checking condition (4.3) on the elements of a basis so that we can be sure that it holds for the whole finite dimensional space. To this end, we discretise by choosing basis functions $\{\phi_i : \Omega \to \mathbb{R}\}_{i=1}^M \subset \mathcal{H}^1_0(\Omega)$ and work in the space which they span; call this space S. This allows us to write the discrete form of Equation (4.3) in which we seek a (discrete) $u_{N,hp} \in S$ such that

$$(\nabla u_{N,hp}, \nabla\chi) = \langle g_N, \chi\rangle_{\partial\Omega_N} - (\nabla u_D, \nabla\chi) \qquad \forall\chi \in S. \tag{4.4}$$

Writing $u_{N,hp} = \sum_{i=1}^M u_i\phi_i$ for scalars u_i which we are to determine, this can be re-cast as the problem: *find* $u_i,\ i = 1, 2, \ldots, M$ *such that*

$$\sum_{i=1}^{M} u_i\,(\nabla\phi_i, \nabla\phi_j) = \langle g_N, \phi_j\rangle_{\partial\Omega_N} - (\nabla u_D, \nabla\phi_j) \qquad \forall j \in \{1, 2, \ldots, M\}\,.$$

We now replace u_D by a discrete approximation $u_{D,hp} = \sum_{i=M+1}^{N} u_i \phi_i$ using additional basis functions $\{\phi_i\}_{i=M+1}^{N}$ which are non-zero on the Dirichlet boundary, giving

$$\sum_{i=1}^{M} u_i \left(\nabla\phi_i, \nabla\phi_j\right) = \left\langle g_N, \phi_j \right\rangle_{\partial\Omega_N} - \sum_{i=M+1}^{N} u_i \left(\nabla\phi_i, \nabla\phi_j\right) \quad \forall j \in \{1, 2, \ldots, M\}, \tag{4.5}$$

where there are no unknowns on the right-hand side. This allows us to express our approximate solution $u_{hp} = u_{N,hp} + u_{D,hp}$ to Problem (4.1) as an element of a finite dimensional subspace $S_D := span\left\{\phi_i\right\}_{i=1}^{N} \cap \mathcal{H}_D^1(\Omega)$.

Remark. *When we introduced u_D, we allowed for it to be any element of $\mathcal{H}_D^1(\Omega)$. Because $u_{D,hp} \in S_D \subset \mathcal{H}_D^1(\Omega)$, $u_{D,hp}$ would have been a perfectly good choice for u_D in the first place, except for the fact that it may not satisfy the Dirichlet boundary conditions exactly. Thus $u_{D,hp}$ is an approximation only in the sense that it may not be equal to g_D on $\partial\Omega_D$.*

We now need to define some basis functions; in order to do this we first explain the discretisation of the domain Ω into the small pieces (elements) which will provide a structure upon which finite element basis functions ϕ_i for the Galerkin method can be defined.

4.1.2 Finite Element Meshes

A *mesh* for the purposes of finite elements is a finite set $\mathcal{M}$ of *elements* ϵ_i, which are open subsets of Ω with the property $\cup_i \overline{\epsilon_i} = \overline{\Omega} \subset \mathcal{R}^l$ and such that when $i \neq j$, if $\overline{\epsilon_i} \cap \overline{\epsilon_j} =: e_{ij}$ is non-empty it may only be of a restricted type depending on the dimensionality and type of elements used, corresponding to requiring that the elements tile Ω in a suitably "nice" manner. For example, in this work we use tetrahedral elements when working in three dimensions; in this case we only allow the intersection e_{ij} to be a whole triangular face, a whole edge line-segment, or a vertex of *both* ϵ_i and of ϵ_j. Similarly, in two dimensions we use triangular elements and e_{ij} must be a shared vertex or complete shared edge of both triangles, and e_{ij} can only be single points for the line-segment elements in one-

dimensional space. Associated with the mesh is the set of nodes $\mathcal{N}$, which consists of all points which are vertices of at least one element.

Let $h_i = diam(\epsilon_i) := \sup\{d(x,y)|x,y \in \epsilon_i\}$, where $d(\cdot,\cdot)$ is the standard Euclidean metric on $\mathbb{R}^l$, be the diameter of ϵ_i, $h = max_i\{h_i\}$, $\rho_{\epsilon_i} = sup\{diam(B)|B \text{ a ball in } \epsilon_i\}$, and Q and R independent of $\mathcal{M}$. We work with meshes that satisfy for all i

$$\frac{h}{h_i} \leq Q, \quad \frac{h_i}{\rho_{\epsilon_i}} \leq R, \tag{4.6}$$

which are the properties of quasiuniformity of the mesh and non-degeneracy of its elements, respectively [78]. We suppose during this work that we have such a mesh for the domain Ω under consideration.

4.1.3 The Reference Element

Design and implementation of a finite element method is simplified greatly by the introduction of an abstract reference element, E. This is essentially a stand-in for each and every element of the mesh, with no physical coordinates or meaning on its own. For E in one, two and three dimensions we respectively use $[0,1]$, the triangle with vertices $\{(0,0),(0,1),(1,0)\}$ and the tetrahedron with vertices $\{(0,0,0),(1,0,0),(0,1,0),(0,0,1)\}$; these are shown in Figure 4.2. When we wish to define properties or structure which are needed on every element of the mesh, we can simply do so on the reference element. In dimensions one, two and three, our reference elements are a line segment, a triangle and a tetrahedron, respectively. Each physical mesh element ϵ_i is then additionally equipped with a (linear) reference map $r_i : \overline{\epsilon_i} \to E$ which describes precisely how it is related to the reference element. In order to avoid "turning elements inside out", we demand that $det\,(jac(r_i)) > 0$. We use E as a domain for the N_{loc} reference basis function pieces $\left\{\hat{\phi}_k : E \to \mathbb{R}\right\}_{k=1}^{N_{loc}}$ which are required to define the basis on a single element. Thus, if we wish to construct a basis function ϕ_j whose support contains the physical element ϵ_i, then on that element it can be written in the form $\phi_j|_{\epsilon_i} := \hat{\phi}_k \circ r_i$ for some $k \in \{1,2,\ldots,N_{loc}\}$.

This approach means that even though we may need millions of basis functions for the whole finite element system, we only need to explicitly define N_{loc} pieces of them; the basis functions in the actual domain are constructed from these using the reference maps r_i. Determining the value of N_{loc} is complicated by the problem of matching basis functions in neighbouring elements; in the high-order case it can be larger than an initial calculation based upon local function space dimensionality would suggest; we return to this point in Section 4.5.1.

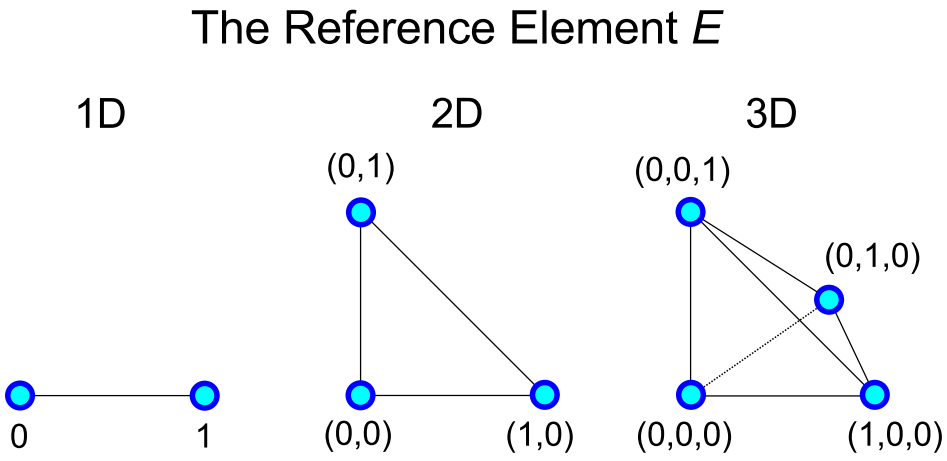


Figure 4.2: The reference element E in each dimensionality.

Remark. *The reference maps r_i need not be linear; by using non-linear maps we can construct elements with curved surfaces in order to better capture the epicardium, endocardium or myocardial microstructure. This would be useful for constructing coarse but detailed meshes. Such maps define* isoparametric elements [100].

4.1.4 The Linear Basis $p = 1$

Conceptionally, *nodal bases* are the simplest to think about. For N_n such basis functions, $\{\phi_i\}_{i=1}^{N_n}$, there are N_n points $x_i \in \Omega$ with the property that each ϕ_i has an x_i associated with it in the sense that $\phi_i(x_j) = \delta_{ij}$, where δ_{ij} is the Kronecker delta function, meaning that if $u = \sum_{i=1}^{N} u_i\phi_i$, $u(x_i) = u_i$. This gives the nice property that we can easily read off the value of $u(x_i)$ by simply looking at the basis function coefficients u_i. The most

common choice here is to use linear finite elements, which are defined by their satisfaction of these properties using the vertices $\mathcal{N}$ of the mesh as the associated points, together with a requirement for them to be linear on each element. Note that this definition forces ϕ_i to be continuous on Ω and satisfies the requirement that the support of each basis function be small; the latter is required so that as many of the scalar products as possible in Equation (4.5) are zero, thus simplifying (increasing the sparsity of) the linear system that it defines. We refer to this as the $p = 1$ basis in this work.

There are other bases which find use in the finite element method; the one that we shall need in this work is the *hierarchical basis* which will be introduced later. Note that hierarchical and nodal bases agree when the polynomial degree is 1, but they differ for higher degrees.

4.1.5 Completing the Discretisation of Laplace's Equation using $p = 1$ Linear Finite Elements

Now that we have defined a basis, we can examine Equation (4.5) more closely. We begin by considering the $p = 1$ basis for S. Because functions in S take the value zero on the Dirichlet portion of the boundary, the basis functions whose associated nodes lie on this boundary must be excluded from its basis. Suppose that M of the N mesh nodes do not lie on $\partial\Omega_D$, and that the ϕ_i are indexed such that the basis functions with nodes x_i lying on $\partial\Omega_D$ are exactly those with $i > M$, then we can easily find values for the u_i with $i > M$ in Problem (4.5), allowing us to re-write it as: *find* $u_i,\ i = 1, 2, \ldots, M$ *such that*

$$\sum_{i=1}^{M} u_i \left(\nabla\phi_i, \nabla\phi_j\right) = \langle g_N, \phi_j \rangle_{\partial\Omega_N} - \sum_{i=M+1}^{N} g_D(x_i) \left(\nabla\phi_i, \nabla\phi_j\right) \quad \forall j \in \{1, 2, \ldots, M\}, \tag{4.7}$$

which is a system of M linear equations in M unknowns, so we can re-write it as

$$\boldsymbol{A}\boldsymbol{u} = \boldsymbol{f}, \tag{4.8}$$

where the stiffness matrix $\boldsymbol{A}$ is such that $\boldsymbol{A}_{ij} = (\nabla\phi_i, \nabla\phi_j)$, $1 \leq i, j \leq M$, $\boldsymbol{u} = (u_1, u_2, \ldots, u_N)^T$, and $\boldsymbol{f}$ is a vector with components

$$f_j = \langle g_N, \phi_j \rangle_{\partial\Omega_N} - \sum_{i=M+1}^{N} g_D(x_i) \, (\nabla\phi_i, \nabla\phi_j) \, .$$

Upon solving this system for $\boldsymbol{u}$, we obtain the approximation $u_{hp} = u_{N,hp} + u_{D,hp}$ in the piecewise linear space $S_D \subset \mathcal{H}^1_D(\Omega)$ by appending to $\boldsymbol{u}$ the vector

$$(g_D(x_{M+1}), g_D(x_{M+2}), \ldots, g_D(x_N)) \, ;$$

this is then the complete vector of coefficients for the basis functions $\{\phi_i\}_{i=1}^N$ which describes the finite element solution u_{hp}.

Remark. *Because we have taken a discrete form of the function u_D, u_{hp} will only satisfy the Dirichlet boundary conditions approximately unless the original g_D is the trace of a function in the finite element space S_D.*

Solving systems such as that given by Equation (4.8) is a computationally demanding problem to which there are many possible approaches; selecting the correct method is vital for efficiency. We do not say more about this here, but will return to it in Section 4.6. We do pause here to note that M is typically very large due to the small elements which are often required to properly represent the boundaries of the domain Ω and to achieve accuracy with the $p = 1$ basis; clearly an S_D built on a mesh with small elements will contain better approximations to the true solution than one built on a mesh with large elements using the same basis on each element, see Figure 4.3; this figure also sketches how high-order basis functions give better approximation properties, which we shall expand on in Section 4.5. Our choice of basis functions with small support means that many of the entries in $\boldsymbol{A}$ are zero, and while the matrix can be very large, it is sparse. Without this property, many problems discretised using finite elements would be computationally intractable due to excessive memory requirements.

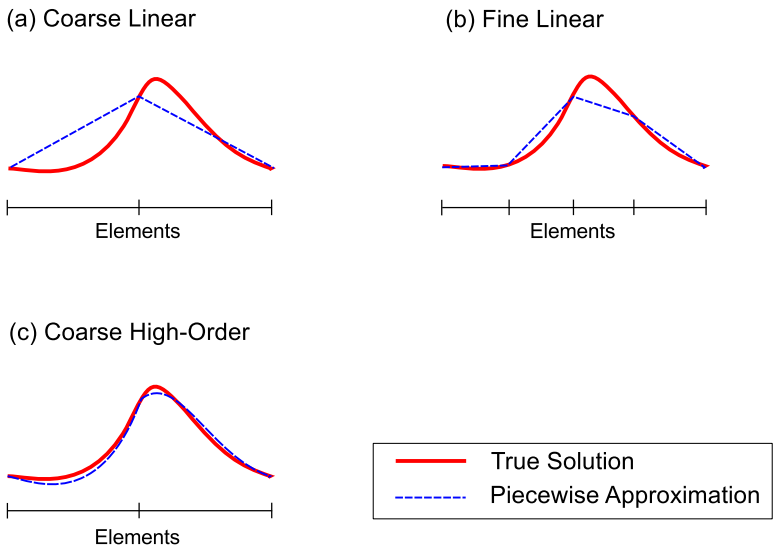


Figure 4.3: Sketch of how approximation power can be increased by working with a finer mesh or with higher-order polynomial elements.

4.2 The Heat Equation

So far we have examined the application of the finite element method to a stationary problem, but the monodomain problem in which we are interested is a time-dependent non-linear problem. We work now with a slightly more complex example, as it allows us to explore the treatment of the time derivative without the added complication of the non-linear forcing term which occurs in the monodomain equation.

The heat equation problem is: *given* $T > 0$, *for* $t \in [0, T]$ *find* $u(x, t)$ *such that*

$$\frac{\partial u}{\partial t} = \Delta u \text{ in } \Omega, \tag{4.9a}$$

$$\hat{n} \cdot \nabla u = 0 \text{ on } \partial\Omega, \tag{4.9b}$$

$$u(x, 0) = u^0(x) \text{ in } \Omega, \tag{4.9c}$$

with initial condition $u^0(x)$. We have chosen to work with simpler homogeneous natural (Neumann) boundary conditions on the entire boundary in order to avoid unnecessary complication. Laplace's equation can be seen as a special case of the heat equation where the temperature distribution in Ω has reached equilibrium, provided that we use the same

boundary conditions.

The finite element discretisation of System (4.9) proceeds in the same manner as for System (4.1), although we work here with $\mathcal{H}^1(\Omega)$ because we have Neumann boundary condition data for the whole boundary; the fact that the FEM automatically deals with such boundary conditions without having to resort to tweaking the function spaces is the reason that Neumann boundary conditions are also known as *natural* boundary conditions in finite element theory. We define

$$L^2\left(0,T;X\right) := \left\{ w : (0,T) \to X \;\middle|\; w \text{ is measurable and } \int_0^T \|w(t)\|_X^2 \, dt < \infty \right\},$$

where $\|\cdot\|_X$ is the norm on X [113]. Beginning with the weak formulation of the problem defined by System (4.9): *given* $T > 0$, *find* $u \in L^2\left(0,T;\mathcal{H}^2(\Omega)\right)$ *such that* $\forall\ t \in [0,T]$,

$$(u_t, \chi) = (\Delta u, \chi) \qquad \forall \chi \in \mathcal{H}^1(\Omega), \tag{4.10}$$

we need to choose a method of treating the time derivative. In this work, we do so using Euler difference quotients, and so we make the approximation $u_t(x, t+\Delta t) \approx \frac{u(x,t+\Delta t)-u(x,t)}{\Delta t}$ for some constant time-step Δt; consistent with the initial condition (4.9c), we write $u^0(x) = u(x,0)$ and $u^n(x)$ for the temporally-semidiscrete approximation to $u(x, n\Delta t)$, and our numerical scheme will yield approximations at times $t = 0,\ \Delta t,\ 2\Delta t,\ \ldots,\ T$. Inserting this difference quotient and assuming that Δt divides T, Equation (4.10) becomes

for $n = 1,\ 2,\ \ldots,\ \frac{T}{\Delta t}$, *find* $u^n \in \mathcal{H}^2(\Omega)$ *such that*

$$(u^n, \chi) - \Delta t\,(\Delta u^n, \chi) = \left(u^{n-1}, \chi\right) \qquad \forall \chi \in \mathcal{H}^1(\Omega), \tag{4.11}$$

which uses an implicit time-stepping scheme. Under this discretisation, solving this problem on a single time-step is reduced to approximating an elliptic problem with a forcing term given by the state of the system on the previous time-step.

Applying Green's theorem to Equation (4.11), removing the $\mathcal{H}^2(\Omega)$ requirement and re-arranging, we have *for* $n = 1,\ 2,\ \ldots,\ \frac{T}{\Delta t}$, *find* $u^n \in \mathcal{H}^1(\Omega)$ *such that*

$$(u^n, \chi) + \Delta t\,(\nabla u^n, \nabla \chi) = \left(u^{n-1}, \chi\right) \qquad \forall \chi \in \mathcal{H}^1(\Omega),$$

where the zero-derivative condition (4.9b) on the whole boundary means that we have no new terms arising due to boundary conditions.

Now, using a set of basis functions $\{\phi_i : \Omega \to \mathbb{R}\}_{i=1}^N$ such that $span\,\{\phi_i : 1 \leq i \leq N\} = S \subset \mathcal{H}^1(\Omega)$, we can expand in this basis the approximation u_{hp}^n to u^n that we wish to determine, with the subscript hp indicating that this form is now spatially discrete. We assume additionally that the spatially discrete form u_{hp}^{n-1} of u^{n-1} is given in this same basis; this is reasonable because we will have used the same method[1] to generate u_{hp}^{n-1}, with the exception being u^0 which may need to be projected into S before beginning simulation. Thus, writing $u_{hp}^n = \sum_{i=1}^N u_i^n \phi_i(x)$ and seeking unknown real scalars u_i^n, analogously to Equation (4.7) our fully discrete problem reads: *for* $n = 1,\ 2,\ \ldots,\ \frac{T}{\Delta t}$, *find* u_i^n *for* $i = 1,\ 2,\ \ldots, N$ *such that*

$$\sum_{i=1}^N u_i^n\,(\phi_i, \phi_j) + \Delta t \sum_{i=1}^N u_i^n\,(\nabla\phi_i, \nabla\phi_j) = \sum_{i=1}^N u_i^{n-1}\,(\phi_i, \phi_j) \qquad \forall j \in \{1, 2, \ldots, N\},$$

which leads to the matrix system: *for* $n = 1,\ 2,\ \ldots,\ \frac{T}{\Delta t}$, *find* $\boldsymbol{u}^n$ such that

$$\boldsymbol{M}\boldsymbol{u}^n + \Delta t \boldsymbol{A}\boldsymbol{u}^n = \boldsymbol{M}\boldsymbol{u}^{n-1}, \tag{4.12}$$

where $\boldsymbol{M}$ is the mass matrix and is defined by $\boldsymbol{M}_{ij} = (\phi_i, \phi_j)$, $\boldsymbol{A}$ is as defined in the Section 4.1, and $\boldsymbol{u}^n = (u_1^n, u_2^n, \ldots, u_N^n)$ is the vector of basis function coefficients for

[1] In the spatially-adaptive that we explore later in this work, the basis varies from time-step to time-step. In that case, we deal with this by representing the forcing term $\boldsymbol{u}^{n-1}$ using the the full maximal basis that the adaptivity has available to it, with zeros in place of the coefficients of any absent basis functions. In this way, the previous system state is always fully included. Note that this requires the copy of $\boldsymbol{M}$ on the right-hand side of Equation (4.12) to be rectangular.

time-step n. Here, we are integrating the right-hand side using $\boldsymbol{M}$; this is referred to as *matrix-based assembly* and is important for efficiency.

4.3 A Simple Reaction-Diffusion System: Fisher's Equation

Fisher's equation [114] is a simple example of a reaction-diffusion equation which takes us another step towards the monodomain system. It models the spread of a species through a (one-dimensional) habitat. Reaction-diffusion equations are parabolic PDEs which are characterised by processes on two levels; the diffusion process which affects the global behaviour of the solution (in the case of Fisher's equation this corresponds to species migration), and the reaction process which affects the solution only locally (here, logistic birth and death).

A nondimensionalised form of Fisher's Equation is: *for* $t \in [0, T]$, *find* $u(x, t)$ *such that*

$$\frac{\partial u}{\partial t} = \Delta u + u(1 - u) \text{ in } \Omega \tag{4.13a}$$

$$\hat{n} \cdot \nabla u = 0 \text{ on } \partial\Omega \tag{4.13b}$$

$$u(x, 0) = u^0 \text{ in } \Omega, \tag{4.13c}$$

which can be fully discretised using the same techniques as described above to obtain $\forall j \in \{1, 2, \ldots, N\}$,

$$\sum_{i=1}^{N} u_i^n \left(\phi_i, \phi_j\right) + \Delta t \sum_{i=1}^{N} u_i^n \left(\nabla\phi_i, \nabla\phi_j\right) = \sum_{i=1}^{N} u_i^{n-1} \left(\phi_i, \phi_j\right) + \Delta t \left(u^{n-1}(1 - u^{n-1}), \phi_j\right),$$

where we have chosen to treat the equation semi-implicitly in time, using the values at time-step $n - 1$ in the final term on the right.

The interesting complexity of Fisher's compared to the heat equation is that there

is an additional forcing term which is non-linear; this can not be represented using the $\{\phi_j\}_{j=1}^N$ as it no longer lies in S. This means that we can not integrate the forcing term exactly using $\boldsymbol{M}$ as we did for the heat equation in Equation (4.12). An alternative would be to compute the integrals on the right-hand side using numerical quadrature; however, this can become very computationally demanding as the size of the domain and the number of elements in the mesh grows, so we wish to avoid using quadrature if possible. However, we can use matrix-based assembly here, but we must be aware of the error that this introduces. Specifically, if we work with the $p = 1$ basis and for each j we write

$$\begin{aligned}\left(u^{n-1}(1-u^{n-1}), \phi_j\right) &\approx \sum_{i=1}^{N} u_i^{n-1}(1-u_i^{n-1})\left(\phi_i, \phi_j\right) \\ &= \boldsymbol{M}\left(\boldsymbol{u}^{n-1} - \boldsymbol{u}^{n-1} \odot \boldsymbol{u}^{n-1}\right), \end{aligned} \tag{4.14}$$

where we use the operator $\odot$ to mean pairwise multiplication of corresponding elements of the two vectors to produce an elementwise-squared vector, we see that what we are actually integrating is not the actual forcing term, but rather a piecewise linear interpolant of it which is accurate at the nodes of the mesh; this is a projection into the finite element space. We will return to this point when we look at the nonlinear transmembrane current term which appears in the monodomain equation. Note for now that if we were using a basis that could represent quadratic functions, but were still computing solutions using only the linears, we would be able to integrate this forcing term exactly by finding appropriate basis coefficients. With the $p = 1$ basis however, the final discretisation of System (4.13) is

$$\left(\boldsymbol{M} + \Delta t \boldsymbol{A}\right)\boldsymbol{u}^n = \boldsymbol{M}\left(\boldsymbol{u}^{n-1} + \Delta t\left(\boldsymbol{u}^{n-1} - \boldsymbol{u}^{n-1} \odot \boldsymbol{u}^{n-1}\right)\right), \tag{4.15}$$

which we can solve for for each n.

More details on FEM can be found for example in [100, 115].

4.4 Linearisation and Time Discretisation

We have chosen here to use a first-order semi-implicit time-stepping scheme, but there are other options. We could, for example, have used a standard Forward Euler scheme by taking the value of the diffusion term from the previous ($n-1$-th) time-step, resulting in the analogue

$$\boldsymbol{M}\boldsymbol{u}^n = ((1+\Delta t)\boldsymbol{M} - \Delta t\boldsymbol{A})\,\boldsymbol{u}^{n-1} - \Delta t\boldsymbol{M}\left(\boldsymbol{u}^{n-1}\odot\boldsymbol{u}^{n-1}\right)$$

of Equation (4.15). This would result in a scheme with the same asymptotic accuracy, but which was less stable. Alternatively, we could have chosen to work with a fully implicit scheme; one in which all the terms outside of the time derivative are evaluated at the new time-step with index n. In this case, the analogue of Equation (4.15) is

$$((1-\Delta t)\boldsymbol{M} + \Delta t\boldsymbol{A})\,\boldsymbol{u}^n = \boldsymbol{M}\left(\boldsymbol{u}^{n-1} - \Delta t\boldsymbol{u}^n\odot\boldsymbol{u}^n\right)$$

The advantage of this would be that it generally provides even greater numerical stability, but the price of this is that we can no longer solve the system of equations by algebraic manipulation alone; additionally we must use a costly iterative procedure such as Newton's method [116, 117, 69]. This is done by writing

$$F(\boldsymbol{u}^n) = ((1-\Delta t)\boldsymbol{M} + \Delta t\boldsymbol{A})\,\boldsymbol{u}^n - \boldsymbol{M}\left(\boldsymbol{u}^{n-1} - \Delta t\boldsymbol{u}^n\odot\boldsymbol{u}^n\right),$$

and solving the equation $F(\boldsymbol{u}^n)=0$ for $\boldsymbol{u}^n$ using the iterative Newton scheme

$$\boldsymbol{u}^n_{k+1} = \boldsymbol{u}^n_k - J_F^{-1}(\boldsymbol{u}^n_k)F(\boldsymbol{u}^n_k),$$

where J_F is the Jacobian of F and the indices k denote the iterate number. Whether or not this is beneficial is dependent on whether accuracy or stability is limiting the size of Δt; if it is stability, then the more expensive Newton iterations may be worthwhile [118].

These time-stepping schemes are all first-order accurate and as such are simple to implement and computationally cheap. Therefore, they are natural first-attempt choices for the time discretisation, and in practice are reasonably good choices for our problem. For a more advanced time-stepping scheme, it has been suggested that the second-order accurate schemes Crank-Nicholson / Adams-Bashforth and second order semi-implicit backwards differentiation are excellent choices due to the much larger time-step that they allow and improved accuracy that they provide [69]. While it is important to ensure that we have accuracy in both space and in time, we see from our numerical results for the monodomain in Table 5.2 that the spatial and temporal errors are not strongly interconnected; when we fix h and p and reduce the time-step size by an order of magnitude to $\Delta t = 0.001\ ms$, there is a relatively uniform shift in the CV across all but the coarsest, lowest order simulations. Thus we see that we can make meaningful accuracy comparisons for the monodomain across different values of h and p and using a fixed $\Delta t = 0.01\ ms$ in the simple semi-implicit scheme of the type used in Equation (4.15), but must remember that with this time discretisation, a non-insignificant temporal error remains. When using such high-order numerical schemes in applications which require high accuracy, this temporal error must be reduced by using a smaller time-step or more accurate time integration method. However, justified by the observed approximate independence of the spatial and temporal errors, and because the focus of this work is on spatial accuracy, we do not concern ourselves with more advanced or efficient time-stepping methods, but note that the present work could make a good starting point for an approach which also involves reducing or controlling the temporal error, by using either a more accurate or an adaptive time integration method.

4.5 Finite Element Spaces

In Section 4.1.4 we described the the $p = 1$ piecewise-linear finite element basis which is very widely used in practice [23, 24, 22, 25], and in Section 4.2 we refrained from

working with a particular basis, preferring to leave the choice open. In both cases, we were defining a finite-dimensional space S in which we were to find approximate solutions to our equations. In this work, we use specific high-order bases, and we describe them now.

The label $p = 1$ for the piecewise-linear basis refers to the fact that the basis functions, and thus also the generated finite element approximation, are polynomials of degree at most one on each element of the mesh $\mathcal{M}$. This nomenclature is indicative of the families of basis functions that we are to define now; our high-order finite element spaces will be piecewise-polynomial of degree $p \geq 1$ supported on the elements of $\mathcal{M}$, and continuous in Ω.

For $p \geq 1$, the subspace S is determined in the following way. We select an $\mathcal{M}$ satisfying Inequalities (4.6) with quasiuniform element diameter h and a basis $\{\phi_i\}_{i=1}^N$ of continuous piecewise polynomials on each element of degree up to and including p in each element, and denote the resulting subspace by S_{hp}; this is our maximal solution space. There is no unique way of doing this, neither in terms of specific basis functions nor general properties of the basis, but we choose the space S_{hp} which can be obtained from the standard linear FEM space S_{h1} by hierarchically adding basis functions of increasing degree to each element up to degree p, so that an arbitrary polynomial on Ω of degree p lies in S_{hp} [119]. This choice of basis has the computationally advantageous property of *hierarchy* such that on each element the basis of degree p is the same as the basis of degree $p+1$ with the degree $p+1$ functions removed. We will need this property when we consider adaptivity in Chapter 6.

For example, with $p = 4$ on a 1D reference element $[0, 1]$ we have (scaled forms of) the basis functions

$$x, \quad 1-x, \quad x\,(1-x) \quad x\,(1-x)\left(\frac{1}{2}-x\right) \text{ and } x\,(1-x)\left(\frac{1}{3}-x\right)\left(\frac{2}{3}-x\right)$$

which are the two components of a linear basis function, the quadratic, the cubic and the

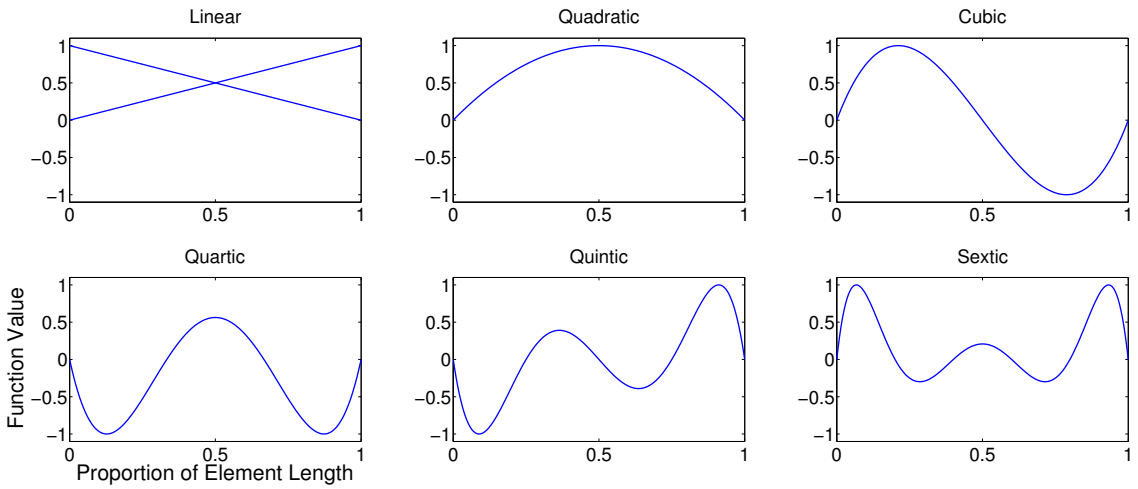


Figure 4.4: The basis functions p=1-6 in one dimension, shown here on an element of diameter 1.

quartic, respectively; these clearly satisfy the property of hierarchy. The one-dimensional basis functions up to degree p=6 are shown in Figure 4.4. In 2D with $p = 4$, there are three linear, three quadratic, four cubic basis and five quartic basis functions partially or wholly supported on each element; see Figures 4.5 and 4.6 for some of these. A complete list of the basis functions used in this work are given in Appendix A.

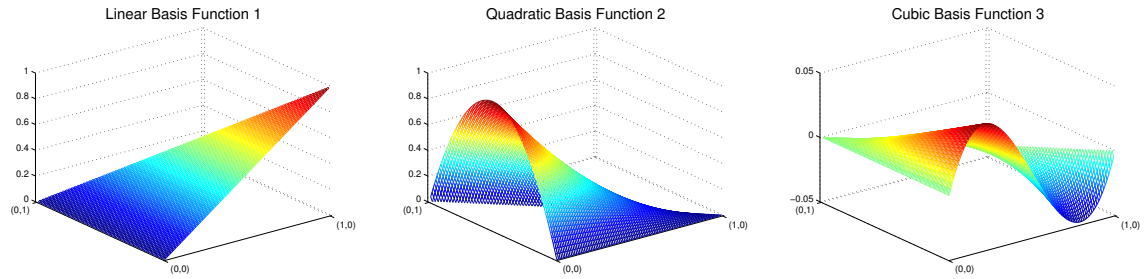


Figure 4.5: Three of the fifteen hierarchical basis function pieces required in 2D on the reference element for a quartic finite element approximation of the solution. Note that when mapped to the real mesh from the reference element, each basis function piece is generally just part of one of the basis functions that is actually supported on multiple elements; the exceptions are the "bubble" functions which are zero on the whole boundary of the element, see Figure 4.6.

An example of a basis without the hierarchy property is the degree-p nodal basis, defined by the requirements that each of the N basis functions ϕ_i has an associated node

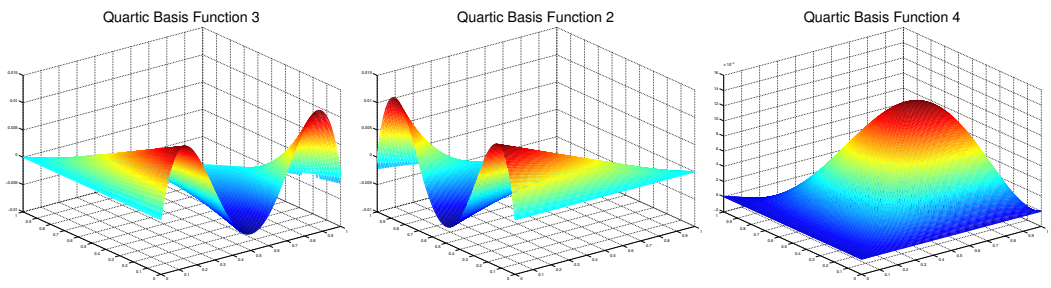


Figure 4.6: Three of the fifteen hierarchical basis function pieces required in 2D on the reference element for a quartic finite element approximation of the solution. Note that when mapped to the real mesh from the reference element, each basis function piece is generally just part of one of the basis functions that is actually supported on multiple elements; here the first two plots would match over an element edge to form the two components of a complete basis function, and the third plot shows a "bubble" function which is zero on the whole boundary and so is supported on only one element.

x_i such that $\phi_i(x_j) = \delta_{ij}$, and that the space S_{hp} as defined above is spanned. Clearly, there must be more nodes here than just those of $\mathcal{N}$; for example, with $p = 2$ in in two dimensions, we use the six nodes shown in Figure 4.7. Nodal bases have the nice property that at the nodes we can read off the value of the solution by simply looking at the computed coefficient of the associated basis function, but since we need the hierarchy property for the p-adaptivity that we will investigate in Chapter 6, we can not use a nodal basis to represent the transmembrane potential field in computations. We will need such a high-order nodal basis in an auxiliary role to treat the spatial discretisation of the non-diffusing cell state variables; these are given by the vector PDE in Equation (1.1b). Thus we only need to know that nodal bases exist; we only use their nodal values rather than their complete functional forms. Unless noted otherwise, in this work we will always be thinking in terms of hierarchical bases.

4.5.1 Revisiting the Reference Element

Now that we have introduced our higher-order bases, we examine how these relate to the reference element E. Basis function pieces are defined on the reference element and transformed by the reference maps r_i as $\hat{\phi}_k \circ r_i : \epsilon_i \to \mathbb{R}$ to the mesh element ϵ_i. The

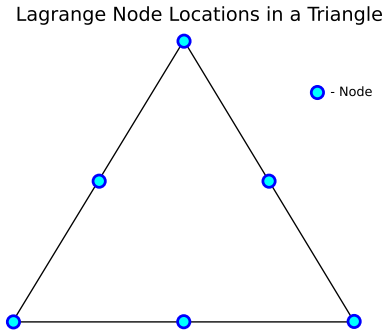


Figure 4.7: Locations of the linear (mesh-nodal) and quadratic (side-midpoint) nodes for a Lagrange p=2 basis on a triangle in 2D.

resulting functions are generally not complete basis functions in themselves, they are just the part of a basis function which is supported on ϵ_i; any part of the basis function which lies in a neighbouring element must be built using an appropriate $\hat{\phi}_k$ which ensures continuity across their shared boundary. Thus, as in Section 4.1.3, we have a family of what can be thought of as basis function building blocks $\hat{\phi}_k$ defined on E. We must piece these together in order to construct whole basis functions on the mesh.

An issue arises with basis functions that lack appropriate symmetry, so that it is not immediately clear which basis function pieces should be used on two neighbouring elements in order to ensure continuity across their shared boundary. Cubic basis functions in 2D such as the cubic depicted in figure 4.5 provide an example of this; the following explanation of the problem is also presented graphically in Figure 4.8. We consider an element ϵ_i and a basis function associated with an edge $\hat{\gamma}$ of E, $\hat{\phi}_7$, the first of the four cubics in 2D, similar to the cubic depicted in Figure 4.5. Suppose we have constructed part of a basis function using this $\hat{\phi}_7$; let this be $\hat{\phi}_7 \circ r_i$ and let $\gamma = \overline{\epsilon_i} \cap \overline{\epsilon_j}$ be the associated mesh edge. Suppose then that $r_j(\gamma) = \hat{\gamma}$ also, then it appears at first that we need to use $\hat{\phi}_7$ again to form the complete basis function. This becomes a problem however, because we are now attempting to match a function with a rotation of itself along γ (see Figure 4.8), and that function is not symmetrical about the mid-point of γ. The answer is to use $-\hat{\phi}_7$ in one of the two cases. Identifying when to do this across the whole mesh requires careful book-keeping. This negative form can be thought of as another reference basis

function in its own right as opposed to one derived from $\hat{\phi}_7$, resulting in a greater number of reference basis functions $\hat{\phi}_i$ being defined on E than the dimensionality of S_{hp} on an element. This kind of thinking was hinted at in Section 4.1.3 in comments regarding the value N_{loc}, and really comes into its own when considering basis function matching on element boundaries in 3D, as this case is far more complicated.

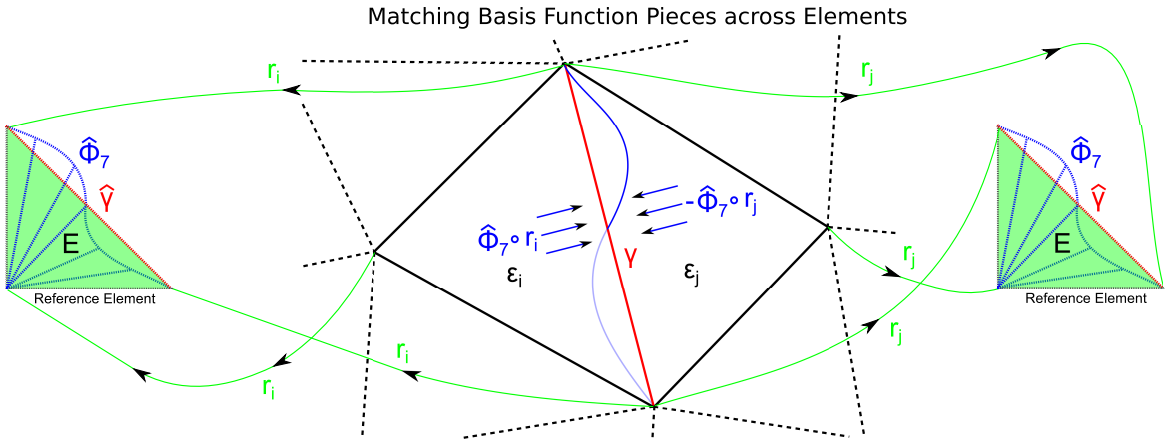


Figure 4.8: Two elements ϵ_i and ϵ_j with a shared boundary γ. The central blue curve shows the value of the cubic basis function on γ, which is the trace on γ of $\hat{\phi}_7 \circ r_i : \epsilon_i \to \mathbb{R}$, and also the trace on γ of $-\hat{\phi}_7 \circ r_j : \epsilon_j \to \mathbb{R}$, showing that the basis function constructed from these two components is continuous on γ, as required. Notice how the reference maps r carry the reference element to ϵ_i and ϵ_j in such a way that $\hat{\phi}_7$ is required twice in the construction of the complete basis function; the requirement for continuity is satisfied by using $-\hat{\phi}_7 \circ r_j$, which is an inversion of the $\hat{\phi}_7$ depicted on the reference element.

4.5.2 Approximation Properties of the Finite Element Method

In this section, for simplicity we assume that g_D is precisely the trace of a function which lies in S_D. The reason why the FEM gives high-quality approximate solutions to PDEs can be seen in the property of Galerkin orthogonality. For the Laplacian problem, this is can be seen by taking Equations (4.3) and (4.4); with $u = u_N + u_D$ and $u_{hp} = u_{N,hp} + u_D$, subtracting one of these from the other gives

$$(\nabla(u - u_{hp}), \nabla\chi) = 0 \quad \forall \chi \in S, \tag{4.16}$$

which demonstrates that u_{hp} is the closest point in S_D to u as measured in the norm induced by the inner product $(\nabla\cdot, \nabla\cdot)$. To see this, we note that $a(f,g) := (\nabla f, \nabla g)$ is an inner product on $\mathcal{H}_0^1(\Omega)$, and write $\|f\|_a := \sqrt{a(f,f)}$ for the norm it induces; this is a full norm because the Poincaré Inequality [118] can be applied in $\mathcal{H}_0^1(\Omega)$ due to the fixed-zero boundary values in this space. We then have

$$\begin{aligned} \|u - u_{hp}\|_a^2 &= a(u - u_{hp}, u - u_{hp}) \\ \text{(by Equation (4.16))} \qquad &= a(u - u_{hp}, u - \chi) \quad \forall \chi \in S_D \\ \text{(Cauchy-Schwarz)} \qquad &\leq \|u - u_{hp}\|_a \|u - \chi\|_a , \end{aligned}$$

which shows the closest-point property by giving us [100]

$$\|u - u_{hp}\|_a \leq \|u - \chi\|_a \qquad \forall \chi \in S_D. \tag{4.18}$$

This property affords us a nice way of thinking about the advantages of high-order finite element spaces S_{hp}, which were introduced in Section 4.5. They have the property $S_{h1} \subsetneq S_{h2} \subsetneq \cdots \subsetneq S_{hp} \subsetneq \mathcal{H}^1(\Omega)$; we see that by increasing p we increase the size of the subspace within which we are obtaining the closest point to the weak solution u. Although this argument gives us no guarantee of convergence or even of improvement, it does provide us with an intuitive feel for the benefit of increasing p. Given Inequality (4.18), we can be more precise about how this error behaves as p increases if we can find a particular element $\chi \in S_D$ for which we know something about the value of $\|u - \chi\|_a$. In order to do this, we can turn to polynomial interpolation theory [100, 120], which tells us how well a polynomial of degree p can approximate an arbitrary function, so long as it has a particular minimum degree of smoothness. Specifically, Babuska and Suri [78] show the following.

Lemma 4.5.1. *Let $k > \frac{3}{2}$ such that $u \in \mathcal{H}^k(\Omega)$. Then there exists an element $\mathcal{I}u \in S_D$ and $C > 0$ which depends on k and on Q and R in (4.6) but is independent of u, p and*

h such that for $\mu = min\{k, p+1\}$,

$$\|u - \mathcal{I}u\|_{1,2} \leq Ch^{\mu-1}p^{k-1}\|u\|_{k,2}.$$

Combining this result with Inequality (4.18) and using the fact that $\|u-\chi\|_a \leq \|u-\chi\|_{1,2}$, we see that

$$\|u - u_{hp}\|_a \leq Ch^{\mu-1}p^{k-1}\|u\|_{k,2},$$

which gives us an asymptotic convergence bound on the error. A similar result can be shown in the case where we have a non-empty Dirichlet boundary [78].

More relevantly for our cardiac application, this same argument regarding the closest point applies to our parabolic equations too. For example, to apply this to Fisher's equation, we can take $a(f,g) := (f,g) + \Delta t(\nabla f, \nabla g)$ and suppose that there was no error in the approximation at the previous time-step by taking $u_{hp}^{n-1} = u^{n-1}$. This ensures that the forcing term on the right-hand side is identical in both the time-semidiscrete case

$$(u^n, \chi) + \Delta t(\nabla u^n, \nabla \chi) = (u^{n-1}, \chi) + \Delta t(u^{n-1}(1-u^{n-1}), \chi) \quad \forall \chi \in \mathcal{H}^1(\Omega)$$

and fully-discrete form

$$(u_{hp}^n, \chi) + \Delta t(\nabla u_{hp}^n, \nabla \chi) = (u_{hp}^{n-1}, \chi) + \Delta t(u_{hp}^{n-1}(1-u_{hp}^{n-1}), \chi) \quad \forall \chi \in S; \tag{4.19}$$

here, we also assume that we integrate the right-hand side of Equation (4.19) without using the efficiency approximation given in Equation (4.14) so that

$$a(u^n - u_{hp}^n, \chi) = 0.$$

This new $a(\cdot,\cdot)$ is a perfectly good inner product, so essentially the same argument as used in the elliptic case demonstrates that, in the norm induced by this inner product, the closest-point property holds. There are much more involved arguments which demon-

strate that convergence occurs; one of these will be given for the cardiac monodomain equation in Section 5.2.1.

4.6 Linear System Preconditioning

Because each basis function has a degree of freedom associated with it, increasing p results in an enlargement of the linear system, such as Equation (4.15), which represents the fully-discrete form of the problem, both in terms of the dimensions of the matrix system and of the matrix fill-in. However, because using a mesh with larger h will result in fewer elements making up Ω and thus fewer degrees of freedom (DOF), we can carefully balance a larger h with a higher $p > 1$ when designing S_{hp}, reducing the overall number of DOF whilst maintaining or even improving numerical accuracy. This is not surprising in the light of Inequality (1.2).

This reduction in the overall DOF is the mechanism by which the high-order finite element method delivers its efficiency improvements. Because the linear systems used for realistic cardiac electrophysiology simulations are typically very large (millions of DOF), direct linear solvers such as Gaussian elimination are uncompetitive; we must instead use iterative Krylov methods such as preconditioned conjugate gradients (PCG) [121]. Because conjugate gradients will require more iterations to converge to a solution when the condition number of the system matrix is larger, this number is highly relevant to numerical efficiency. The computational cost of solving a linear system iteratively increases with both the number of DOF and the number of iterations required for convergence, and increasing p will cause a corresponding increase in the condition number, so we must choose an effective preconditioner to bring the iteration count down.

Let us write our linear system as $\boldsymbol{A}x = b$, where $\boldsymbol{A}$ is an $N \times N$ sparse matrix. Solving for x using PCG will give an exact solution to the linear system after N iterations; this is still computationally demanding. However, typically after significantly fewer iterations it will produce a result which is acceptably close to the exact solution. The number of

iterations required is related to the *condition number* $\kappa(\boldsymbol{A}) \in [1, \infty)$ of $\boldsymbol{A}$, which is the ratio of the largest to smallest eigenvalue of $\boldsymbol{A}$ [121]. Generally, the larger $\kappa(\boldsymbol{A})$, the slower the iterative convergence. Note that a good preconditioner will cause a large reduction in the condition number, and the condition number will be affected by the mesh and the degree of the polynomial basis (larger degrees increase the condition number) used in the FEM discretisation. We can reduce the condition number using a preconditioner $\boldsymbol{P}$ as follows.

(Left-) preconditioning of a linear system involves the application of some linear operator $\boldsymbol{P^{-1}}$ to $\boldsymbol{A}$ followed by performing conjugate gradient iteration on the preconditioned system $\boldsymbol{P^{-1}A}x = \boldsymbol{P^{-1}}b$, so that the relevant condition number is $\kappa(\boldsymbol{P}^{-1}\boldsymbol{A})$. This $\boldsymbol{P^{-1}}$ can be thought of as some approximate inverse of $\boldsymbol{A}$; the condition number of the identity matrix is 1, so we wish to choose $\boldsymbol{P}$ such that $\boldsymbol{P^{-1}A}$ is close to the identity matrix so that $\kappa(\boldsymbol{P}^{-1}\boldsymbol{A})$ will be small. Adjusting the condition number in this manner is necessary because it is usually large for FEM system matrices, particularly when $p > 1$.

The requirements for a good preconditioner $\boldsymbol{P^{-1}}$ are as follows.

1. $\boldsymbol{P^{-1}A}$ has a significantly smaller condition number than $\boldsymbol{A}$.
2. $\boldsymbol{P^{-1}}y$ is computationally cheap to evaluate.
3. The memory storage requirements for $\boldsymbol{P^{-1}}$ are not too great.
4. The iteration counts should be largely independent of the discretisation values h and p.
5. Ideally, the evaluation of $\boldsymbol{P^{-1}}x$ should scale well to many processors in parallel.

These requirements can be difficult to reconcile with one another; for example, the greatest reductions in the condition number will generally require the most memory to store the preconditioner. Thus it can be quite difficult to make absolute statements about which preconditioning method is best, as this is affected by available memory, processor distribution and even the simulation protocol itself [13]. Additionally, the ease of implementation

of different preconditioners varies from the incredibly simple to the extremely complex; this affects the choices made in designing software. Efficiency comparisons are possible between simulations which use the same preconditioner, but are not straightforward when using different preconditioners because the optimal choice is situation-dependent, making general statements difficult.

4.6.1 ILU(0) Preconditioner

In our one-dimensional and fixed-order two-dimensional simulation studies, we will use an incomplete LU with zero fill-in (ILU(0)) preconditioner for our linear system. This is a very simple preconditioner which consists of approximate lower $\boldsymbol{L}$ and upper triangular $\boldsymbol{U}$ factors of the system matrix $\boldsymbol{A}$ so that $\boldsymbol{A} \approx \boldsymbol{LU}$, but adjusted so that the factors have the same sparsity pattern as $\boldsymbol{A}$ within the nonzero triangle. The triangular nature of this preconditioner means that it can be applied cheaply. It provides an effective way of reducing the condition number for p=1 FEM, but it can be expected to perform less well in terms of the condition number of the preconditioned system as the ratio of the time-step Δt to h^2 increases [122, 14], and its usefulness decreases further with larger p.

4.6.2 Block Preconditioner

In Chapter 6, for our two-dimensional adaptive simulations, we employ a block-preconditioning approach which is related to ILU(0). The precise details are best described together with the adaptive scheme, so we leave those to Section 6.4.1, but we provide an overview here.

Because we arrange the indexing of our degrees of freedom so that $\boldsymbol{A}$ can be decomposed into blocks as

$$\boldsymbol{A} = \begin{pmatrix} \boldsymbol{A}_{LL} & \boldsymbol{A}_{LH}^T \\ \boldsymbol{A}_{LH} & \boldsymbol{A}_{HH} \end{pmatrix},$$

where the subscripts L or H indicate whether linear or high-order (or both) basis function interactions occur within the block. We exploit this structure to design a preconditioner

which approximately inverts the blocks $\boldsymbol{A}_{LL}$ and $\boldsymbol{A}_{HH}$ using their ILU(0) decompositions; the advantage of doing this in the adaptive setting is that we can adjust the preconditioner for the $\boldsymbol{A}_{HH}$ block without involving that for the never-changing $\boldsymbol{A}_{LL}$, reducing the work involved in performing an adapt. It also displays better performance than ILU(0) in reducing the condition number for $p > 1$.

4.6.3 Additive Schwarz Preconditioner

For all our 3D studies, we use an overlapping block-Jacobi additive Schwarz preconditioner, chosen to allow us to evaluate the performance of our method with the sort of preconditioner that would be a good choice for whole-heart scale practical simulation on a large-scale parallel architecture.

For a linear system of the form $\boldsymbol{A}x = b$, the idea behind additive Schwarz preconditioners [100] is that we can approximate the solution of a linear system arising from the discretisation of a PDE in a domain Ω by breaking that domain up into small, slightly overlapping subdomains $\{\Omega_k\}_{k=1}^{K}$ with $\Omega = \cup_{k=1}^{K}\Omega_k$ and $\Omega_k = \cup_{i\in I_k}\epsilon_i$ for some indexing sets I_k, finding some approximate local form $\boldsymbol{A}_k$ of the operator $\boldsymbol{A}$ on each subdomain Ω_k and solving the problem locally in a subspace $S_k \subset S_{hp}$ on each Ω_k. These local solutions can then be combined in a very simple way to generate an approximate global solution.

We can write down this preconditioner $\boldsymbol{P}_{AS}^{-1}$ as

$$\boldsymbol{P}_{AS}^{-1} = \sum_{k=1}^{K} \hat{R}_k^{-1}\boldsymbol{A}_k R_k \tag{4.20}$$

where the $R_k : S_{hp} \to S_k$ are restriction operators, each to the subspace of the finite element space associated with Ω_k. The operators $\hat{R}_k^{-1}$ can be thought of as the inverses of the R_k; however, because this inverse is not well-defined we should remember that strictly they are not inverses, but rather operators that embed the S_k in S_{hp} in the obvious way given that $S_k \subset S_{hp}$.

The choice of subspace that we make is guided by, but differs slightly from, the method

of Pavarino [123]. Our number of subdomains K is equal to the number of nodes of the mesh, and our k^{th} local subspace is defined using the k^{th} linear basis function ϕ_k as $S_k := span\,(B_k)$ with $B_k := \{\phi_j \mid supp(\phi_k) \cap supp(\phi_j) \neq \emptyset\}$. Our subdomains Ω_k are therefore given by $\Omega_k = supp\left(\sum_{\phi_j \in B_k} \phi_j\right)$ and are small patches of elements around the k^{th} mesh node; see Figure 4.9.

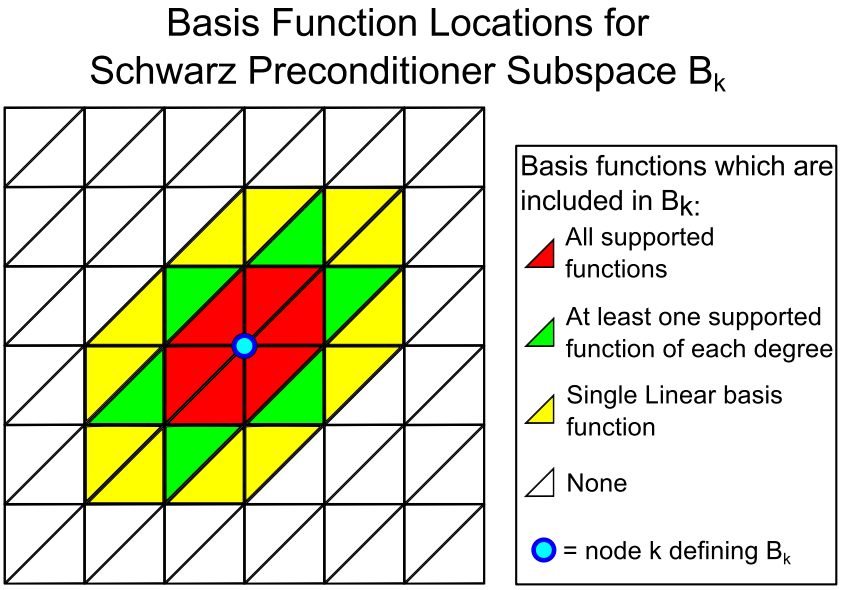


Figure 4.9: The patch of elements which support basis functions which are included in the local subspace B_k associated with the k^{th} mesh node and linear basis function for the additive Schwarz preconditioner.

The advantage of this preconditioner is that it is immediately parallelisable, and very likely will scale excellently; the study of Giraud et al. [124] indicates that superlinear parallel scaling can occur for such preconditioners. The terms of the sum in Equation (4.20) can be distributed to as many processors as we wish, until the workload of each processor is too small to make doing so worthwhile; the result then needs to be gathered back together and summed to complete the application of the preconditioner. We will see in Section 5.5 that the additive Schwarz preconditioner is effective for controlling the condition number across all values of p investigated.

4.7 Chapter Conclusions

We have seen how the finite element method is applied to PDEs, and noted that nonlinearity in the forcing terms can introduce numerical error when using matrix-based assembly. We have described hierarchical and nodal finite element bases in general terms, and noted that we will require both for discretising the monodomain equation. We have seen that we must be careful to ensure we have preserved continuity across element boundaries when defining a hierarchical basis using the reference element, and we have discussed the requirements for and examples of linear system preconditioners. This sets us up with everything we need in order to apply these methods to the monodomain system in the next chapter. A major reference for the current chapter is [77].

Chapter 5

Application to the Monodomain System

Having seen high-order FEM in a more general setting, we specialise to the problem that we are interested in: the monodomain system. In this chapter, we shall see how to discretise it and perform simulation using uniform-degree high-order FEM. Via an a priori *estimate, we shall understand how the errors in the transmembrane potential and in the cell model are intertwined so that both affect the accuracy of the FEM solution for the transmembrane potential.*

Using norms of the error in one dimension, we show experimentally that the scheme provides the high accuracy that we expect, in agreement with the a priori *estimate. Simulations in one, two and three dimensions across a range of different domains and conductivity tensor fields show that we obtain high-accuracy and improved efficiency over state-of-the-art $p = 1$ FEM. We include simulations using a 3D test-problem which has been used in the literature to evaluate the accuracy of a number of electrophysiology simulation packages [58].*

5.1 The Monodomain Equation

Recall the monodomain system with a cell model which we focus on in this work is written as: *for $t \in [0, T]$, find the time-dependent scalar field $u(x, t)$ and the time-dependent vector*

field $w(x,t)$ *such that*

$$C_m \frac{\partial u}{\partial t} - \frac{1}{\beta}\nabla \cdot (\sigma \nabla u) - I_{ionic}(u,w) = I_{stim}(x,t) \text{ in } \Omega, \tag{5.1a}$$

$$\frac{\partial w}{\partial t} - g(u,w) = 0 \text{ in } \Omega, \tag{5.1b}$$

$$u(x,0) = u^0(x) \ \forall x \in \Omega, \tag{5.1c}$$

$$\hat{n} \cdot (\sigma \nabla u) = 0 \text{ on } \partial\Omega, \tag{5.1d}$$

$$w(x,0) = w^0(x) \ \forall x \in \Omega. \tag{5.1e}$$

This describes the propagating waves in the transmembrane potential $u(x,t)$ which coordinate contraction of cardiac myocytes [6]. As previously, x is a point in the l-dimensional myocardial domain Ω, Ω has boundary $\partial\Omega$ which is often polygonal due to the methods used to generate it from cardiac MRI [95], $\hat{n}$ is the outward-pointing unit surface normal to $\partial\Omega$, u^0 and w^0 are the initial conditions, and we have time t, cell membrane capacitance C_m, cell surface area to volume ratio β, and current $I_{total}(u,w,x,t) = I_{ionic}(u,w) + I_{stim}(x,t)$, consisting of the transmembrane ionic current $I_{ionic}(u,w)$ as described by the cell model and the stimulus current $I_{stim}(x,t)$ as determined by the experimental protocol. g describes how the m non-diffusing cell model state variables $w(x,t) = (w_1(x,t), \ldots, w_m(x,t))^T$ vary in time. $\sigma(x) \in \mathbb{R}^{l \times l}$ is the conductivity tensor, which is symmetric, satisfies $a^T \sigma(x) a > 0 \ \forall a \in \mathbb{R}^l \setminus \{0\}$ (positive definiteness), and its eigenvalues give the electrical conductivity in the direction of each associated eigenvector. In this work, in 3D we use the ten Tusscher 2006 cell model [57], and in 2D, the Luo-Rudy Phase I (LR91) cell model [34], modified according to [49], for g and I_{ionic}; these quantities describe how the cardiac cells locally affect the transmembrane potential u. We cast the transmembrane potential PDE from system (5.1) into the weak form: *find* $u \in L^2\left(0,T; \mathcal{H}^1(\Omega) \cap \mathcal{H}^2(\Omega)\right)$ *such that for all* $\chi \in \mathcal{H}^1(\Omega)$ *and* $\forall\ t \in [0,T]$,

$$(u_t, \chi) - \frac{1}{\beta C_m}\left(\nabla \cdot (\sigma \nabla u), \chi\right) - \frac{1}{C_m}\left(I_{total}, \chi\right) = 0. \tag{5.2}$$

Integrating by parts and taking advantage of the removal of the second-order derivative, we adjust the problem statement to be *find* $u \in L^2\left(0,T;\mathcal{H}^1(\Omega)\right)$ *such that* $\forall\ t \in [0,T]$

$$(u_t,\chi) - \frac{1}{\beta C_m}\left(\int_{\partial\Omega} \chi\sigma\nabla u\cdot\hat{n}\ dx^{l-1} - (\sigma\nabla u,\nabla\chi)\right) - \frac{1}{C_m}(I_{total},\chi) = 0\ \forall\ \chi\in\mathcal{H}^1(\Omega)$$

where $\hat{n}$ is an outward-pointing unit surface normal. Note that a sufficiently smooth solution to this will also be a solution to Equation 5.2. Applying the boundary condition (5.1d) and relaxing the second-order weak differentiability requirement on u gives the problem *find* $u \in L^2\left(0,T;\mathcal{H}^1(\Omega)\right)$ *such that* $\forall\ t \in [0,T]$

$$(u_t,\chi) + \frac{1}{\beta C_m}(\sigma\nabla u,\nabla\chi) - \frac{1}{C_m}(I_{total},\chi) = 0\ \forall\ \chi\in\mathcal{H}^1(\Omega).$$

Choosing some finite-dimensional subspace $S\subset\mathcal{H}^1\left(\Omega\right)$ to work in, with basis $\{\phi_i\}_{i=1}^N$, we can find a spatially discrete approximation $u_d = \sum_{i=1}^N u_i\phi_i$ to u by determining appropriate basis function coefficients u_i. Inserting this, we obtain the semi-discrete system of equations as follows:

$$\left(\sum_{i=1}^N u_{t,i}\phi_i,\phi_j\right) + \frac{1}{\beta C_m}\left(\sigma\nabla\left(\sum_{i=1}^N u_i\phi_i\right),\nabla\phi_j\right) - \frac{1}{C_m}\left(I_{total},\phi_j\right) = 0 \quad j=1,\ 2,\ldots,\ N.$$

We now write $\boldsymbol{u} = (u_1,\ u_2,\ldots,\ u_N)^T$, $\tilde{\boldsymbol{I}}_{total} = ((I_{total},\phi_1),\ (I_{total},\phi_2),\ldots,(I_{total},\phi_N))$, and $\boldsymbol{M}$ and $\boldsymbol{A}$ for the matrices with (j,i)th entry (ϕ_i,ϕ_j) and $(\sigma\nabla\phi_i,\nabla\phi_j)$, the mass and stiffness matrices respectively, to obtain

$$\boldsymbol{M}\boldsymbol{u}_t + \frac{1}{\beta C_m}\boldsymbol{A}\boldsymbol{u} - \frac{1}{C_m}\tilde{\boldsymbol{I}}_{total} = 0, \tag{5.3}$$

where we must remember that this is still nonlinear because $\tilde{\boldsymbol{I}}_{total}$ is still a function of $\boldsymbol{u}$ and w; we show how to treat this in Section 5.2.2. We note that we have not yet discretised this equation in time; this is secondary to our objective of designing an

efficient spatial method, and we will come to this also in Section 5.2.2. For now, where we require the concept of a fully-discrete approximation to u using a time-step Δt and some time-discretisation scheme, we will write

$$u^n = u^n(x) = \sum_{i=1}^{N} u_i(n\Delta t)\phi_i(x),$$

where the coefficients u_i are defined on the discrete set of time points $\{0, \Delta t, 2\Delta t, \dots\}$, and depend on the time-stepping scheme that we choose.

Remark. *We use the notation $\tilde{\boldsymbol{I}}_{total}$ here to distinguish from the form $\boldsymbol{I}_{total}$ that we will introduce later. The former includes the current I_{total} in the matrix equation (5.3) as $\tilde{\boldsymbol{I}}_{total} := ((I_{total}, \phi_1),\ (I_{total}, \phi_2), \dots, (I_{total}, \phi_N))^T$, with I_{total} not necessarily polynomial. The latter will approximate $\tilde{\boldsymbol{I}}_{total}$ by $((\Pi I_{total}, \phi_1),\ (\Pi I_{total}, \phi_2), \dots, (\Pi I_{total}, \phi_N))^T = \boldsymbol{M}\boldsymbol{I}_{total}$, with ΠI_{total} a projection of I_{total} into the finite element space, and $\boldsymbol{I}_{total} = (I_1, I_2, \dots, I_N)$ the vector of coefficients in the expansion $\Pi I_{total} = \sum_{i=1}^{N} I_i\phi_i$. The projection will be discussed further in Section 5.2.*

5.1.1 Discretising the Cell Model, w

As we shall see in what follows, and as we hinted in Section 4.5, choosing to discretise w using order-$\tilde{p}$ finite elements in space is necessary in order to achieve the high-accuracy that we expect from the scheme; the lower of the degrees used for the discretisation of u and w will define the convergence rate of the scheme with h-refinement, and more importantly, the magnitude of the error on a particular mesh. We explain the discretisation now.

We take the PDE (5.1b) for w and discretise it in each of its components using high order finite elements; starting from the weak form, for each i,

$$\left(\frac{\partial w_i}{\partial t}, \chi\right) = (g_i(u, w_i), \chi) \qquad \forall \chi \in L^2(\Omega),$$

which is discretised in time using a (semi-implicit) Euler scheme as

$$\left(w_i^{n+1} - w_i^n, \chi\right) = \Delta t \left(g_i(u^n, w_i^n, w_i^{n+1}), \chi\right) \qquad \forall \chi \in L^2(\Omega), \tag{5.4}$$

where we use n to denote the time-step to which each quantity belongs. The extra arguments of g in Equation (5.4) represent the fact that we treat terms implicitly wherever possible, taking u^{n+1} and w_i^{n+1} on the right-hand side wherever simple algebraic manipulation allows us to do so without requiring an iterative scheme to perform a time-step update.

As in Section 4.3, we want to be able to write the term on g_i the right-hand side of Equation (5.4) in terms of our basis $\{\phi_j\}$ so that we can perform the right-hand side integration using the mass matrix $\boldsymbol{M}_{cell}$ that we are to construct. We choose to use a nodal basis of degree $\tilde{p}$ for the discretisation and thus for $\boldsymbol{M}_{cell}$; the tilde is used to distinguish from the degree used for approximating the transmembrane potential PDE (5.1a). The need for this distinction will become clear in when we prove Theorem 5.2.1. We have the problem that g will most likely lie outside of the solution space $S_{h\tilde{p}}$ that this basis spans, so we will require a projection Π into our finite element space for the cell model, and must accept the error that this introduces if we do not wish to introduce additional computational expense. Performing this projection means that we are actually solving the equation

$$\left(w_i^{n+1} - w_i^n, \chi\right) = \Delta t \left(\Pi g_i(u^n, w_i^n, u^{n+1}, w_i^{n+1}), \chi\right) \qquad \forall \chi \in L^2(\Omega)$$

as opposed to our original formulation. Expanding using the nodal basis and working in the finite dimensional space $S_{h\tilde{p}}$, we obtain the linear system

$$\boldsymbol{M}_{cell} \boldsymbol{w}_i^{n+1} = \boldsymbol{M}_{cell} \left(\boldsymbol{w}_i^n + \Delta t \ \boldsymbol{\Pi} \boldsymbol{g}_i\right) \tag{5.5}$$

where we have used bold symbols to denote the vectors of basis function coefficients; for

example $\boldsymbol{\Pi g}_i := \left(\Pi g_i^1, \Pi g_i^2, \ldots, \Pi g_i^{N_{cell}}\right)$, thus representing the function $\Pi g_i(u^n, w_i^n, u^{n+1}, w_i^{n+1})$ by $\sum_{j=1}^{N_{cell}} \Pi g_i^j \phi_j$.

Because the matrix $\boldsymbol{M}_{cell}$ is non-singular, this calculation is merely a formality, albeit one worth exposing because it makes an easily-overlooked source of discretisation error apparent. Multiplying on the left by the inverse of this matrix, the cell model problem now completely de-couples into what looks like a system of spatially localised ODEs for each i as

$$\boldsymbol{w}_i^{n+1} = \boldsymbol{w}_i^n + \Delta t \; \boldsymbol{\Pi g}_i; \tag{5.6}$$

these w_i^{n+1} can then be used in the computation of $\boldsymbol{I}_{total}$ to drive the next time-step of the transmembrane potential PDE; see section 5.1.

5.2 The Requirements of Cell Model Treatment in Tissue and an *a priori* Estimate

This Section presents some analysis in support of the cell model discretisation that we choose to use in this work.

We wish to have an approximation to I_{total} of the form

$$I_{total} \approx \sum_{i=1}^{N} I_i \phi_i, \quad \boldsymbol{I}_{total} := (I_{1,}, I_2, \ldots, I_N)^T \in \mathbb{R}^N,$$

so that the integration required to generate the current term in the linear system (5.3) can be reduced to the computationally efficient approximation $\tilde{\boldsymbol{I}}_{total} \approx \boldsymbol{M}\boldsymbol{I}_{total}$, referred to as matrix-based assembly. Because the term I_{ionic} in Equation (5.1a) is nonlinear, in general $I_{total}(u^n, w^n, x, t) \notin S_{hp}$. Thus we require a choice of a suitable projection $\Pi : L^2(\Omega) \to S_{h\tilde{p}} \subset S_{hp}$ for some $\tilde{p}$; we can then work with ΠI_{total}. Additionally, the non-diffusing cell state variables w must be solved with sufficient spatial accuracy; a piecewise-linear approach to w, which is effectively what is used in most implementations,

will limit the overall accuracy of the scheme. We can obtain the accuracy needed by using finite elements of degree $\tilde{p}$ to approximate w; in this case there is a very natural way to construct Π mapping into $S_{h\tilde{p}}$ (see Section 5.2.2), so we take the same $\tilde{p}$ for the degree of the image of Π and the order of the finite elements used for w.

Taking $\tilde{p} = 1$ is not sufficient; the problem with this can be seen by considering the error in 1D simulations at different levels of h-refinement with p kept fixed. We take a 2 cm 1D domain, and perform monodomain simulations with $\tilde{p}$ always set to 1. Define

$$\|h(x,t)\|_{L^\infty(L^2)} = \text{ess sup}_{t\in[0,T]} \left\{ \sqrt{\int_\Omega h(x,t)^2 dx} \right\},$$

where T is the stopping time of the monodomain simulation. For each of $p \in \{1,2,3\}$ and six different values of h in the range $[0.001, 0.01]$ cm (18 simulations in total), we apply a stimulus at one end of the domain, chosen such that it can be applied exactly to all simulations, and run the simulation for $T = 20$ ms to allow the wavefront time to form and propagate. For each simulation, we computed the error in the $L^\infty(L^2)$ norm, using a reference solution generated with $h = 0.0001$ cm and $p = 3$. Figure 5.1 shows the error convergence rates for these simulations, demonstrating the restricted convergence. Because of inequality (1.2) we might expect the error in this norm to be $O(h^{p+1})$, but we only see $O(h^2)$. This is consistent with the quadratic convergence of the error in w caused by $\tilde{p} = 1$. Instead, we let $\tilde{p} = p$ to allow for the full convergence rate. In the next subsection we put this on a solid theoretical foundation.

5.2.1 Analysis for the Error in L^2

We begin with a necessary lemma, supposing that the true solution to system (1.1) has at least k derivatives.

Lemma 5.2.1. *For $u \in H^k(\Omega)$, $\Omega \subset \mathbb{R}^2$, the L^2 projection $\pi : H^k(\Omega) \to S_{hp}$ satisfies*

$$\|u - \pi u\|_{0,2} \leq C h^\mu p^{-k} \|u\|_{k,2}$$

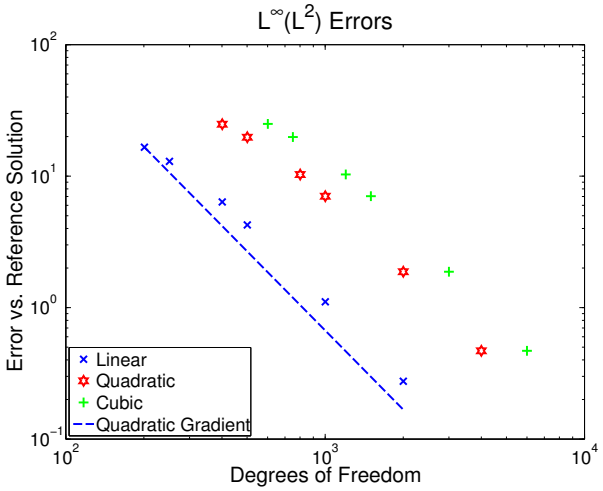


Figure 5.1: 1D simulation results demonstrating that we never do better than quadratic convergence in the $L^\infty(L^2)$ norm when the approximation to w is linear-only, even when the finite element space S_{hp} is of high enough order to allow for better convergence. The points are the measured errors, the dashed line shows a theoretical quadratic convergence gradient. The simulation was 1D and the errors are measured against a reference with mesh spacing h=0.0001 cm, whereas the test simulations use meshes with six different spacings ranging from h=0.01 cm to h=0.001 cm.

for $k > 3/2$ *and* $\mu = min\{p+1, k\}$.

Proof. A modification of Theorem 3 of [125] produces a continuous piecewise polynomial ψ on a mesh of quadrilaterals which satisfies

$$\|u - \psi\|_{0,2} \leq Ch^\mu p^{-k} \|u\|_{k,2};$$

the proof of Theorem 4 of [126] contains the details necessary to modify ψ so that it applies when the mesh consists of triangles. Because π is the L^2 projection, we have $\|u - \pi u\|_{0,2} \leq \|u - \psi\|_{0,2}$. □

The importance of $\tilde{p}$ can be seen via an *a priori* error estimate (see [127]). We examine this in what follows, where we work with the vectorised formulation of system (5.1), treating the spatial discretisation of the non-diffusing state variables in the finite

element framework. To this end we introduce the inner product

$$([a_1\ a_2]^T, [b_1\ b_2]^T)_F := (a_1, b_1) + \sum_{i=1}^{m} (a_{2,i}, b_{2,i})$$

and associated norm $\|\cdot\|_F$, where (a, b) is the standard L^2 inner product, $a_1, b_1 \in L^2(\Omega)$ and $a_2, b_2 \in (L^2(\Omega))^m$. Define

$$\Pi_{h\tilde{p}} : \mathcal{H}^1(\Omega) \times \left(L^2(\Omega)\right)^m \to (S_{h\tilde{p}})^{m+1},$$

the $L^2(\Omega)$ projection into $S_{h\tilde{p}}$ in each of its $m+1$ components, with m the number of components of w, and also the $(m+1) \times (m+1)$ matrix-like operator

$$G := \begin{pmatrix} \nabla & 0 & \dots & 0 \\ 0 & 0 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \dots & \dots & 0 \end{pmatrix}$$

which operates as $G\boldsymbol{u} = G[u\ w]^T = [\nabla u\ 0]^T$.

Dropping the constants and conductivity tensor σ from system (5.1) for clarity, we obtain the weak form of the full system: *find* $\boldsymbol{u} \in L^2\left(0, T; \mathcal{H}^1(\Omega) \times (L^2(\Omega))^m\right)$ *such that* $\forall\ t \in [0, T]$,

$$(\boldsymbol{u}_t, \chi)_F + (G\boldsymbol{u}, G\chi)_F = (\boldsymbol{f}(\boldsymbol{u}), \chi)_F,\ \forall\ \chi \in \mathcal{H}^1(\Omega) \times \left(L^2(\Omega)\right)^m, \tag{5.7}$$

where $\boldsymbol{u}(x,t) := [u(x,t)\ w(x,t)]^T$ and $\boldsymbol{f}(\boldsymbol{u}) := [I_{total}(\boldsymbol{u})\ g(\boldsymbol{u})]^T$, with the symbols as in system (5.1). Let $\boldsymbol{p} = (p, \tilde{p})$, $\boldsymbol{u}_{h\boldsymbol{p}}(x,t) := [u_{hp}(x,t)\ w_{h\tilde{p}}(x,t)]^T$ be the solution to our FEM formulation

$$(\boldsymbol{u}_{h\boldsymbol{p},t}, \chi)_F + (G\boldsymbol{u}_{h,\boldsymbol{p}}, G\chi)_F = (\Pi_{h\tilde{p}}\boldsymbol{f}(\boldsymbol{u}_{h\boldsymbol{p}}), \chi)_F,\ \forall\ \chi \in S_{hp} \times (S_{h\tilde{p}})^m, \tag{5.8}$$

where we are solving for u using elements of order p and for w using elements of order $\tilde{p}$,

and we have applied $\Pi_{h\tilde{p}}$ to obtain an approximation in $S_{h\tilde{p}}$ to the current I_{total} for the purpose of computationally-efficient matrix-based right-hand side assembly of the linear system (5.3).

Let $R_{h\boldsymbol{p}} : \mathcal{H}^1(\Omega) \times (L^2(\Omega))^m \to S_{hp} \times (S_{h\tilde{p}})^m$ be a Ritz-L^2 projection, given by

$$((G + \hat{I})\boldsymbol{u}, (G + \hat{I})\chi)_F = ((G + \hat{I})R_{h\boldsymbol{p}}\boldsymbol{u}, (G + \hat{I})\chi)_F \quad \forall \chi \in S_{hp} \times (S_{h\tilde{p}})^m, \tag{5.9a}$$

$$\int_\Omega (R_{h\boldsymbol{p}}\boldsymbol{u})_1 \, dx^n = 0, \tag{5.9b}$$

where $\hat{I}$ is the $(m+1) \times (m+1)$ identity matrix adjusted so that $\hat{I}(1,1) = 0$, $(z)_1$ denotes the first component of z, and Equation (5.9b) ensures that $R_{h\boldsymbol{p}}$ is well-defined. We note that in particular $R_{h\boldsymbol{p}}$ satisfies

$$(G\boldsymbol{u}, G\chi)_F = (GR_{h\boldsymbol{p}}\boldsymbol{u}, G\chi)_F \quad \forall \chi \in S_{hp} \times (S_{h\tilde{p}})^m. \tag{5.10}$$

Informed by [127], we now prove the following theorem on the error in the finite element approximation, which demonstrates the importance of $\tilde{p}$.

Theorem 5.2.1. *Let $\boldsymbol{u}$ be the solution to system (5.7) with initial conditions as given in system (5.1), $\Omega \subset \mathbb{R}^2$, and $\boldsymbol{u}_{h\boldsymbol{p}}$ the solution to its semidiscrete-in-space form (5.8). Let $\mu = min\{p+1, k\}$ and $\tilde{\mu} = min\{\tilde{p}+1, k\}$, where $\boldsymbol{u}$ has spatial derivatives of order at least k. Suppose $k > 3/2$ and that $\boldsymbol{f}$ is Lipschitz continuous in $\boldsymbol{u}$ with respect to the norm $\|\cdot\|_F$ and with Lipschitz constant L. Then for some constant C, the following* a

priori *estimate for the error at time T holds:*

$$\begin{aligned}\|\boldsymbol{u}_{h\boldsymbol{p}}(T)-\boldsymbol{u}(T)\|_F &\leq C\left(\exp(LT)\left\{\|\boldsymbol{u}_{h\boldsymbol{p}}(0)-\boldsymbol{u}(0)\|_F + h^{\mu}p^{-k}\|u(0)\|_{k,2} + h^{\tilde{\mu}}\tilde{p}^{-k}\sum_{i=1}^{m}\|w_i(0)\|_{k,2}\right.\right.\\ &+ \left[\int_0^T L\left(h^{\mu}p^{-k}\|u\|_{k,2} + h^{\tilde{\mu}}\tilde{p}^{-k}\sum_{i=1}^{m}\|w_i\|_{k,2}\right) + \|(\boldsymbol{f}-\Pi_{h\tilde{p}}\boldsymbol{f})(\boldsymbol{u})\|_F\right.\\ &+ \left.\left. h^{\mu}p^{-k}\|u_t\|_{k,2} + h^{\tilde{\mu}}\tilde{p}^{-k}\sum_{i=1}^{m}\|g_i\|_{k,2}dt\right]\right\} + h^{\mu}p^{-k}\|u(T)\|_{k,2}\\ &+ \left. h^{\tilde{\mu}}\tilde{p}^{-k}\sum_{i=1}^{m}\|w_i(T)\|_{k,2}\right). \end{aligned} \tag{5.11}$$

Proof. Following [127], we decompose the error we wish to bound as

$$\boldsymbol{u}-\boldsymbol{u}_{h\boldsymbol{p}} = \underbrace{\boldsymbol{u}-R_{h\boldsymbol{p}}\boldsymbol{u}}_{\rho} + \underbrace{R_{h\boldsymbol{p}}\boldsymbol{u}-\boldsymbol{u}_{h\boldsymbol{p}}}_{\theta},$$

and then bound ρ and θ separately. θ satisfies

$$\begin{aligned}(\theta_t,\chi)_F + (G\theta,G\chi)_F &= ((R_{h\boldsymbol{p}}\boldsymbol{u})_t,\chi)_F - (\boldsymbol{u}_{h\boldsymbol{p},t},\chi)_F + (GR_{h\boldsymbol{p}}\boldsymbol{u},G\chi)_F\\ &- (G\boldsymbol{u}_{h\boldsymbol{p}},G\chi)_F \quad \forall\chi\in S_{hp}\times(S_{h\tilde{p}})^m;\end{aligned}$$

applying equation (5.10) and rearranging,

$$\begin{aligned}(\theta_t,\chi)_F + (G\theta,G\chi)_F &= -(\boldsymbol{u}_{h\boldsymbol{p},t},\chi)_F - (G\boldsymbol{u}_{h\boldsymbol{p}},G\chi)_F\\ &+ (G\boldsymbol{u},G\chi)_F + ((R_{h\boldsymbol{p}}\boldsymbol{u})_t,\chi)_F \quad \forall\chi\in S_{hp}\times(S_{h\tilde{p}})^m.\end{aligned}$$

The first two terms are replaced using the semidiscrete equation (5.8):

$$\begin{aligned}(\theta_t,\chi)_F + (G\theta,G\chi)_F &= -(\Pi_{h\tilde{p}}\boldsymbol{f}(\boldsymbol{u}_{h\boldsymbol{p}}),\chi)_F + (G\boldsymbol{u},G\chi)_F + ((R_{h\boldsymbol{p}}\boldsymbol{u})_t,\chi)_F\\ &+ (\boldsymbol{f}(\boldsymbol{u}),\chi)_F - (\boldsymbol{f}(\boldsymbol{u}),\chi)_F \quad \forall\chi\in S_{hp}\times(S_{h\tilde{p}})^m.\end{aligned}$$

Using equation (5.7),

$$\begin{aligned}(\theta_t,\chi)_F+(G\theta,G\chi)_F &= (\boldsymbol{f}(\boldsymbol{u})-\Pi_{h\tilde{p}}\boldsymbol{f}(\boldsymbol{u}_{h\boldsymbol{p}}),\chi)_F-(\boldsymbol{u}_t,\chi)_F+((R_{h\boldsymbol{p}}\boldsymbol{u})_t,\chi)_F\\ &= (\boldsymbol{f}(\boldsymbol{u})-\Pi_{h\tilde{p}}\boldsymbol{f}(\boldsymbol{u}_{h\boldsymbol{p}}),\chi)_F+((R_{h\boldsymbol{p}}\boldsymbol{u})_t-\boldsymbol{u}_t,\chi)_F\\ &= (\boldsymbol{f}(\boldsymbol{u})-\Pi_{h\tilde{p}}\boldsymbol{f}(\boldsymbol{u}_{h\boldsymbol{p}}),\chi)_F-(\rho_t,\chi)_F \qquad \forall\chi\in S_{hp}\times(S_{h\tilde{p}})^m\,.\end{aligned}$$

Thus, we have

$$(\theta_t,\chi)_F+(G\theta,G\chi)_F=(\boldsymbol{f}(\boldsymbol{u})-\Pi_{h\tilde{p}}\boldsymbol{f}(\boldsymbol{u}_{h\boldsymbol{p}}),\chi)_F-(\rho_t,\chi)_F \quad \forall\chi\in S_{hp}\times(S_{h\tilde{p}})^m\,. \tag{5.12}$$

Taking $\chi=\theta$ (which is possible because we chose θ such that $\theta\in S_{hp}\times(S_{h\tilde{p}})^m$) and applying Cauchy-Schwarz,

$$(\theta_t,\theta)_F+(G\theta,G\theta)_F=(\boldsymbol{f}(\boldsymbol{u})-\Pi_{h\tilde{p}}\boldsymbol{f}(\boldsymbol{u}_{h\boldsymbol{p}}),\theta)_F-(\rho_t,\theta)_F,$$

$$\frac{1}{2}\frac{d}{dt}\left\|\theta\right\|_F^2+\left\|\nabla\theta_1\right\|_{0,2}^2\le\left(\left\|\boldsymbol{f}(\boldsymbol{u})-\Pi_{h\tilde{p}}\boldsymbol{f}(\boldsymbol{u}_{h\boldsymbol{p}})\right\|_F+\left\|\rho_t\right\|_F\right)\left\|\theta\right\|_F,$$

where θ_1 is the first component of theta. Since $\left\|\nabla\theta_1\right\|_{0,2}^2\ge 0$ and $\frac{d}{dt}\left\|\theta\right\|_F^2=2\left\|\theta\right\|_F\frac{d}{dt}\left\|\theta\right\|_F$, we can write

$$\frac{d}{dt}\left\|\theta\right\|_F\le\left\|\boldsymbol{f}(\boldsymbol{u})-\Pi_{h\tilde{p}}\boldsymbol{f}(\boldsymbol{u}_{h\boldsymbol{p}})\right\|_F+\left\|\rho_t\right\|_F,$$

$$\frac{d}{dt}\left\|\theta\right\|_F\le\left\|\boldsymbol{f}(\boldsymbol{u})-\Pi_{h\tilde{p}}\boldsymbol{f}(\boldsymbol{u})\right\|_F+\left\|\Pi_{h\tilde{p}}\boldsymbol{f}(\boldsymbol{u})-\Pi_{h\tilde{p}}\boldsymbol{f}(\boldsymbol{u}_{h\boldsymbol{p}})\right\|_F+\left\|\rho_t\right\|_F.$$

Using the Lipschitz property of $\boldsymbol{f}$ and the L^2 projection bound $\left\|\Pi_{h\tilde{p}}(\boldsymbol{f}(\boldsymbol{v}))\right\|_F\le\left\|\boldsymbol{f}(\boldsymbol{v})\right\|_F$,

$$\frac{d}{dt}\left\|\theta\right\|_F\le\left\|(\boldsymbol{f}-\Pi_{h\tilde{p}}\boldsymbol{f})(\boldsymbol{u})\right\|_F+L\left\|\boldsymbol{u}-\boldsymbol{u}_{h\boldsymbol{p}}\right\|_F+\left\|\rho_t\right\|_F.$$

Now we can integrate to obtain

$$\left\|\theta(T)\right\|_F\le\left\|\theta(0)\right\|_F+\int_0^T L\left\|\boldsymbol{u}-\boldsymbol{u}_{h\boldsymbol{p}}\right\|_F+\left\|(\boldsymbol{f}-\Pi_{h\tilde{p}}\boldsymbol{f})(\boldsymbol{u})\right\|_F+\left\|\rho_t\right\|_F dt,$$

$$\|\theta(T)\|_F \leq \|\theta(0)\|_F + \int_0^T L\left(\|\theta\|_F + \|\rho\|_F\right) + \|(\boldsymbol{f} - \Pi_{h\tilde{p}}\boldsymbol{f})(\boldsymbol{u})\|_F + \|\rho_t\|_F \, dt.$$

Applying Gronwall's lemma, we see that

$$\|\theta(T)\|_F \leq \exp(LT)\left(\|\theta(0)\|_F + \int_0^T L\|\rho\|_F + \|(\boldsymbol{f} - \Pi_{h\tilde{p}}\boldsymbol{f})(\boldsymbol{u})\|_F + \|\rho_t\|_F \, dt\right).$$

In addition, we have the bounds

$$\begin{aligned} \|\rho(T)\|_F &\leq C\left(h^{\mu}p^{-k}\|u(T)\|_{k,2} + h^{\tilde{\mu}}\tilde{p}^{-k}\sum_{i=1}^{m}\|w_i(T)\|_{k,2}\right) \\ \|\rho_t\|_F &\leq C\left(h^{\mu}p^{-k}\|u_t\|_{k,2} + h^{\tilde{\mu}}\tilde{p}^{-k}\sum_{i=1}^{m}\|g_i\|_{k,2}\right) \end{aligned} \tag{5.13}$$

from [21] and Lemma 5.2.1, where k can be as large as is allowed by the smoothness of u and w. These can be combined with the bound for θ, together with

$$\begin{aligned} \|\theta(0)\|_F &\leq \|\boldsymbol{u}_{h\boldsymbol{p}}(0) - \boldsymbol{u}(0)\|_F + \|R_{h\boldsymbol{p}}\boldsymbol{u}(0) - \boldsymbol{u}(0)\|_F \\ &\leq \|\boldsymbol{u}_{h\boldsymbol{p}}(0) - \boldsymbol{u}(0)\|_F + Ch^{\mu}p^{-k}\|u(0)\|_{k,2} + Ch^{\tilde{\mu}}\tilde{p}^{-k}\sum_{i=1}^{m}\|w_i(0)\|_{k,2}, \end{aligned}$$

which can essentially be found in [127], to arrive at the estimate for the full error:

$$\begin{aligned} \|\boldsymbol{u}_{h\boldsymbol{p}}(T) - \boldsymbol{u}(T)\|_F \leq\ & C\left(\exp(LT)\left\{\|\boldsymbol{u}_{h\boldsymbol{p}}(0) - \boldsymbol{u}(0)\|_F + h^{\mu}p^{-k}\|u(0)\|_{k,2} + h^{\tilde{\mu}}\tilde{p}^{-k}\sum_{i=1}^{m}\|w_i(0)\|_{k,2}\right.\right. \\ &+ \left[\int_0^T L\left(h^{\mu}p^{-k}\|u\|_{k,2} + h^{\tilde{\mu}}\tilde{p}^{-k}\sum_{i=1}^{m}\|w_i\|_{k,2}\right) + \|(\boldsymbol{f} - \Pi_{h\tilde{p}}\boldsymbol{f})(\boldsymbol{u})\|_F\right. \\ &\left.\left. + h^{\mu}p^{-k}\|u_t\|_{k,2} + h^{\tilde{\mu}}\tilde{p}^{-k}\sum_{i=1}^{m}\|g_i\|_{k,2}dt\right]\right\} + h^{\mu}p^{-k}\|u(T)\|_{k,2} \\ &+ h^{\tilde{\mu}}\tilde{p}^{-k}\sum_{i=1}^{m}\|w_i(T)\|_{k,2}\Bigg), \end{aligned}$$

as required. □

The theorem demonstrates that the approximation properties of $S_{h\tilde{p}}$ are crucial. For

example, with $p = 3$ we would like to get a quartic convergence rate of u_{hp} to u at any particular time T in the norm $\|u_{hp} - u\|_{0,2}$ as we refine h, but we can not expect this if $\tilde{p} = 1$ because of the $O(h^2)$ terms that then appear in inequality (5.11). This agrees with our experimental results (see Figure 5.1).

Remark. *The restriction to $\Omega \subset \mathbb{R}^2$ of Theorem 5.2.1 is due to Lemma 5.2.1 only having been proved for $\mathbb{R}^2$. In other dimensionalities, this restriction would be lifted by extending the proof of the Lemma.*

5.2.2 Proper treatment of the cell model PDEs for High Order FEM

Recall that for an element ϵ, a nodal basis $\{\phi_i^L\}_{i=1}^{N(p)}$ of degree p (for the space of polynomials of degree $\leq p$ on ϵ) associated with a set of nodes $\{x_i\}_{i=1}^{N(p)}$, $x_i \in \epsilon$, is such that $\phi_i^L(x_j) = \delta_{ij}\ \forall i, j$ [100]. For our problem (1.1), we use a nodal basis of degree $\tilde{p}$ on the elements of $\mathcal{M}$ to approximate w; the Gauss-Lobatto points associated with polynomials of order $\tilde{p}$ [128] make a good choice here for high-quality approximation. This naturally gives us an approximation ι to $\Pi_{h\tilde{p}}$; we compute the current at each x_i, and we immediately have the nodal basis coefficients for our degree-$\tilde{p}$ projection ιI_{total}, interpolating I_{total} at the points x_i. Informed by Theorem 5.2.1, we take $\tilde{p} \geq p$ to obtain the expected convergence rate for our choice of p.

If we assume that the L^2 projection $\Pi_{h\tilde{p}}$ used in Theorem 5.2.1 and the interpolation operator ι that we replace it by in practice are sufficiently close, we can suppose that the error between $\boldsymbol{f}$ and $\iota\boldsymbol{f}$ is $O\left(h^{\tilde{\mu}}p^{-k}\right)$, where we also assume sufficient smoothness of $\boldsymbol{f}$. Of course, we cannot be certain that this error will be achieved with our scheme given that we have not attempted to carefully approximate $\Pi_{h\tilde{p}}$, but our experimental work has indicated that ι is sufficient.

Because there are no spatial derivatives in the PDE for w, the nodal finite element system reduces to effectively a large number of local ODE systems; these are familiar in

concept as the pointwise ODE models that occur in standard linear FEM approaches to this problem. In practice, this means that we never have to form matrices for the nodal system. As ιI_{total} is represented using the Lagrange basis, but $\boldsymbol{M}$ is the mass matrix in the hierarchical basis, once we have calculated ιI_{total} on each time-step we must change its basis to hierarchical so that we can efficiently integrate the current term as $\boldsymbol{M}\boldsymbol{I}_{total}$.

We discretise System (5.3) in time, with a degree p basis representation of the current $\boldsymbol{I}_{total}$, using (semi-implicit) backward Euler differences with a fixed time-step Δt, and for each function f, we write $f^n_{h\boldsymbol{p}}$ to denote the discretised form of f at time-step n represented using polynomial element degrees $\boldsymbol{p}$ and with element size h. Here, $\boldsymbol{p}$ is a vector and has one entry for each element of the mesh; in this chapter every entry is always p, but this notation will serve us well when we introduce our adaptive algorithm in Chapter 6, where $\boldsymbol{p}$ changes from time-step to time-step and thus should also be considered to be indexed by n. Note that we consider $f^n_{h\boldsymbol{p}}$ to be constant on the interval $[n\Delta t, (n+1)\Delta t)$. The fully-discrete form of System (5.3) that we work with is

$$\left(\boldsymbol{M} + \frac{\Delta t}{\beta C_m}\boldsymbol{A}\right)\boldsymbol{u}^n_{h\boldsymbol{p}} = \boldsymbol{M}\left(\boldsymbol{u}^{n-1}_{h\boldsymbol{p}} + \frac{\Delta t}{C_m}\boldsymbol{I}^{n-1}_{total}\right), \tag{5.14}$$

together with the discrete scheme for the cell model given by Equation (5.6).

5.2.3 Discontinuities in the Cell Model

This subsection describes a deficiency in the cell model which must be overcome for a proper high-order FEM implementation. Many of the cardiac cell models in the literature include some discontinuous functions which describe the voltage-dependent rate at which conductive ion channels embedded in the cell membrane open and close [129]. Such issues can prevent high-order numerical schemes from attaining their theoretical rates of convergence. This has been noted previously for high order temporal schemes [129]; here we identify it as a problem for spatial schemes also. Figure 5.5(a) shows the problem; note the deviation of the quartic and cubic solutions from their respective theoretical

convergence rates. The discontinuity exists because two different analytic expressions have been fitted to experimental data for the voltage-dependent transition rates for the ion channel gates in the cell model. Which of these expressions is used is determined by the transmembrane potential, with the discontinuity at the point where the model switches between them (-40 mV).

A continuous replacement has been around for some time [49] but has not been adopted; the problem has propagated due to the fact that some cell models have been created as modifications of older ones. Introducing this continuous form is required to ensure theoretically optimal errors in the solutions to system (5.1). Compare Figure 5.5(a) with Figure 5.5(b) which differs only in that the cell model used in Figure 5.5(b) has undergone this modification.

Figure 5.2 shows the AP difference between the standard LR91 model and the modified Noble-form LR91; they are quite minor. Given the fact that cell models are generated from experimental data naturally prone to experimental error [130], these differences are probably not worth being concerned with, especially given that the discontinuities do not appear to be biologically justified. We must check for and remove such discontinuities when using a particular cell model for simulation.

5.2.4 Including a Smooth Fibre Field

In one of our test simulations (see Section 5.4.4), we demonstrate applicability of the method when the boundaries of myocardial microstructure must be resolved. We will do this by using a geometry that includes holes representing blood vessels passing through the tissue (see Figure 5.9(b)). In order to construct a realistic conductivity tensor σ we generate fibre orientation vector fields using a Laplace-Dirichlet approach [62]. In order to do this, we solve Laplace's equation on the domain with Dirichlet boundary condition $+1$ on one external edge of Ω, -1 on the opposite external edge and zero Neumann conditions on all other boundaries. Taking the gradient of the resulting scalar field, we obtain a vector field $\boldsymbol{V}(x)$ with streamlines that negotiate around the holes in the domain; we take

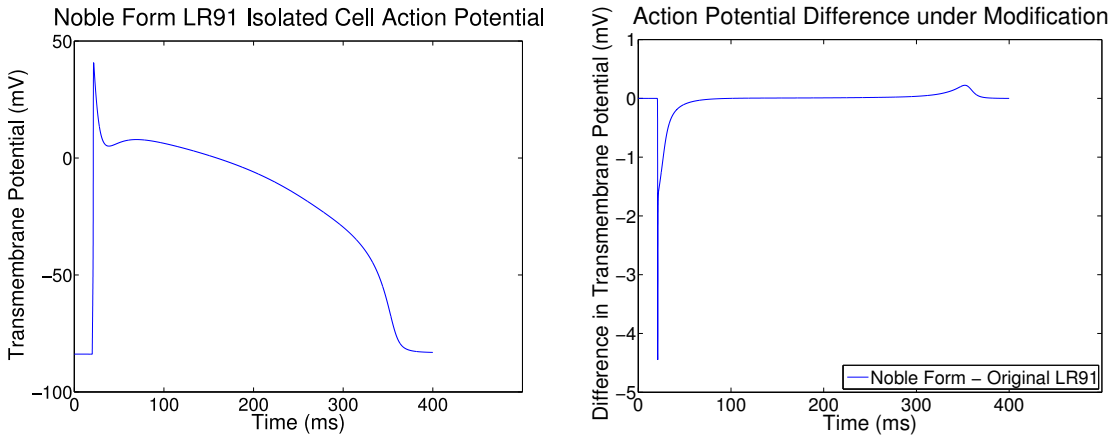


Figure 5.2: Plot comparing the action potential in an isolated cell model using the standard Luo-Rudy 1991 formulation and the Noble-form modification. Left: Noble-form modified LR91 action potential in an isolated cell. On this scale, differences caused by the modification would be hardly noticeable. Right: original LR91 transmembrane potential subtracted from Noble-form LR91 transmembrane potential. Note that the scales on the y-axes differ.

this to be an idealised representation of the manner in which myocardial fibres negotiate their way around blood vessels. At each point x in the domain, we can then construct a 2×2 conductivity tensor $\sigma(x)$ by demanding that it has one eigenvector parallel to $\boldsymbol{V}(x)$ and one eigenvector perpendicular to it, and that the associated eigenvalues are the desired fibre-parallel and fibre-perpendicular conductivity values, respectively. The result is a conductivity tensor field which approximates the way that cardiac fibres negotiate continuously around blood vessels [131].

While this is a smooth field generated as an example, realistic fields derived from imaging data may not be so smooth. A non-smooth fibre field may reduce the convergence rate of the solution with p-refinement; if the fibres have severe discontinuities in some regions, then small elements may be needed to resolve the behaviour there properly, in a way similar to that for fine geometric detail. It is worth noting however that often fibres are prescribed using smoothly-varying functions.

5.3 Notes on Implementation

5.3.1 Cell Model Nodes

Because the monodomain system (5.1) is derived as a homogenisation, there is no concept of a "cardiac cell" in the equations. Instead, we must think of the cell model state variables $w(x)$ as everywhere-defined state fields which evolve as prescribed by the whole monodomain system. These are PDEs which look like ODEs once discretised, as we explained in Section 5.2.2; see also Equation (5.5). However, for the purposes of implementation we need to choose some points at which we compute the values of the cell state variables. We store these locations in the finite element mesh data structure itself, and they are distributed on a typical tetrahedron as shown in Figure 5.3. We then use Lagrange finite elements to approximate the cell model state variables; this effectively means producing a degree-$\tilde{p}$ interpolant of the transmembrane current which agrees with the current at the Lagrange nodes.

Note that most of the cell nodes are on the boundary of the element, and that multiple elements sharing an edge, face or vertex also share the cell nodes that it contains. The former is because the node distribution is constrained by the requirement for continuity across element boundaries of interpolating functions f represented using the associated Lagrange basis. We need to have sufficient nodes on each face or edge to uniquely determine the trace of f there; uniqueness means that elements sharing the boundary must agree there on the value of f. For a degree-$\tilde{p}$ Lagrange basis representation, we use sufficient Lagrange nodes to determine a two-dimensional polynomial of degree-$\tilde{p}$ on each face, and sufficient nodes to determine a one-dimensional degree-$\tilde{p}$ polynomial on each edge. The nodes shared between two elements correspond to the Lagrange basis functions which are supported on both elements, consisting of a piecewise polynomial component in each. The uniqueness means that changing the basis for the current term can be performed in an entirely element-local manner.

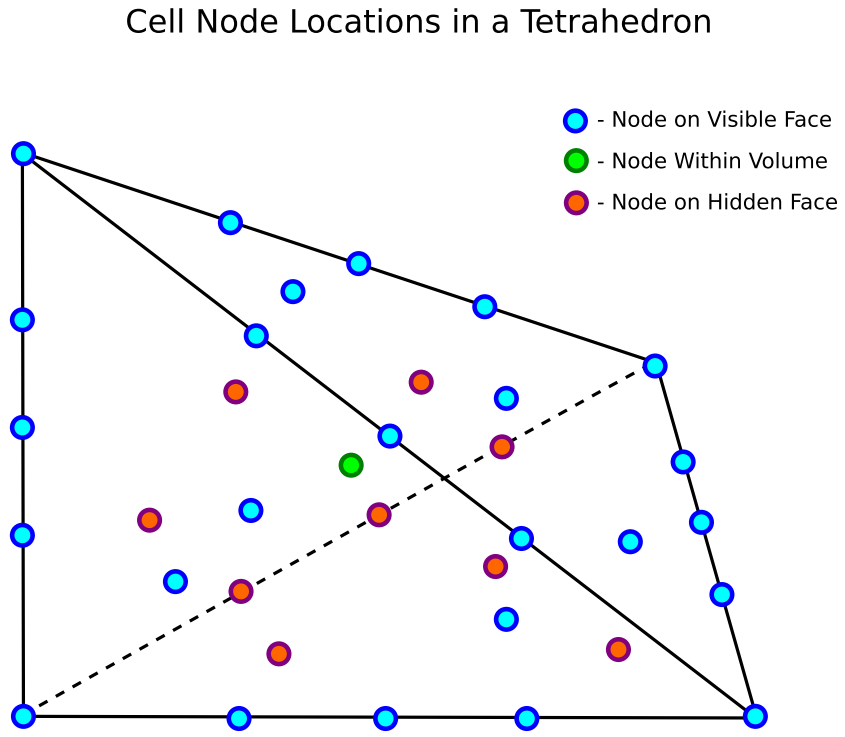


Figure 5.3: The location of the cell model nodes in a typical tetrahedron (i.e. the location of the Lagrange basis nodes). The colours roughly indicate the depth of the point within the 3D volume.

5.3.2 Harnessing Powerful Computer Hardware

The single instruction, multiple data compute capabilities of modern graphics processors (GPUs) means that they can deliver extremely high levels of performance when a problem can be broken down into many instruction-identical calculations. The cell model component of the monodomain system fits this condition perfectly, which means we can boost our simulation speeds considerably by using GPU software. Using the Matlab API Mex [132], we have built an interface between Matlab and the GPU compute language OpenCL [133] which allows us to take advantage of this. Figure 5.4 shows the magnitude of the speed-up that we obtain with OpenCL when compared to single-threaded, compiled Mex code in C. Note that it would take approximately twenty CPU cores to match the speed that we obtain from the GPU. We do not perform any further evaluation of the GPU software that we have designed, but we note that the extra speed has been important in allowing us to perform our larger-scale simulations.

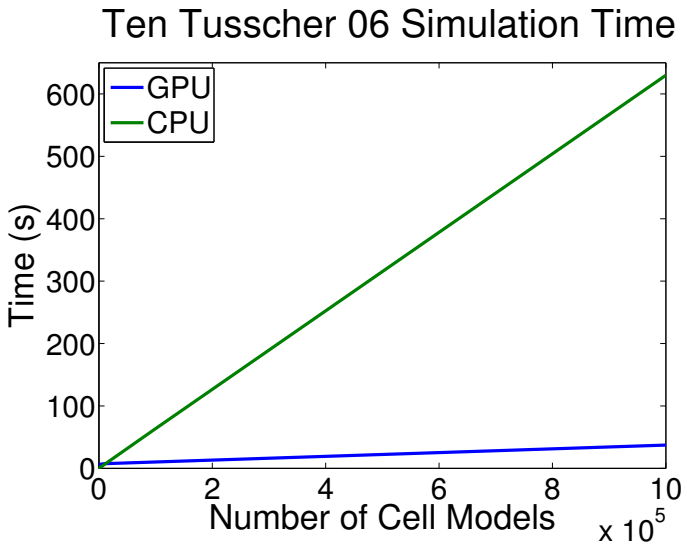


Figure 5.4: Comparison of the wall time required to solve many copies of the ten Tusscher 2006 cell model over 1 ms using either a CPU (3.4 GHz, single thread) or GPU (1,408 stream processors).

5.4 Numerical Experiments with Uniform Degree High-Order Discretisations

We now present the results of using this scheme. All simulations were performed in MATLAB and use a semi-implicit backward Euler time discretisation scheme. We use a fixed $\tilde{p} = 4$ regardless of the value of $p(\leq 4)$ so that we can focus on the effect of varying the approximation order for the transmembrane potential u; leaving $\tilde{p}$ fixed means that we can examine this in a fair manner. Throughout this work, the results tables use highlighted cells to indicate the most interesting data for the attention of the reader. Timings presented are for a 3.4 GHz CPU coupled where necessary with an AMD Radeon 6950 graphics card; the latter is for the easily-parallelised cell model only.

5.4.1 Convergence in 1D

We stimulated a 2 cm 1D domain at one end with a ramp stimulus using a time-step of $\Delta t = 0.001$ ms and simulated the first 20 ms of activation. The errors in two differ-

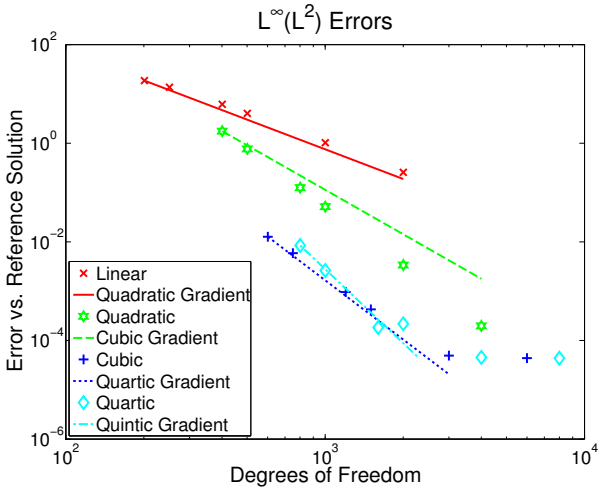
ent norms are presented in Figures 5.5(b) and 5.6(a) and use as a reference a quartic solution generated with h=0.0001 cm. Figure 5.6(a) shows the error measured using the $L^2\left([0,T];\mathcal{H}^1(\Omega)\right)$ norm, for which we expect $O(h^p)$ convergence rates [21]. Note the agreement of Figures 5.5(b) and 5.6(a) with the theoretical error convergence rates presented, and the limited accuracy displayed in Figure 5.5(a) caused by the discontinuities in the standard LR91 cell model. The exponential error convergence rate achievable using p-refinement is emphasised in Figure 5.6(b), and Figure 5.7 shows the same data on semilog axes; in the latter plot we see approximately straight lines, which is as expected for exponential convergence. Note that there is some deviance from the straight lines for large h and large p; we are not certain what has caused this. We use a conductivity of 0.5 Sm^{-1} for all our convergence figures.

In order to perform a robust investigation of conduction velocity (CV), we used a 6 cm domain, stimulated at one end with a ramp stimulus and used $\Delta t = 0.01$ ms; the activation time for the node at the opposite end of the domain and the CVs are presented in Table 5.1. We performed two sets of simulations in order to gather data on the fast and slow conductivities that we will use later, respectively parallel and perpendicular to the fibres in anisotropic 2D simulations. Note how small h needs to be when using linear elements to achieve CV convergence; the error remains as large as 11% (highlighted in red in Table 5.1) with the physiological conductivity σ=0.2 S m^{-1} and state-of-the-art mesh resolution 0.01 cm; consider the effect of even lower conductivities in pathological states. Also highlighted in the table are the coarsest of the linear and high-order results to produce a CV error smaller than 5% (green) and 1% (blue) for each of the two tested conductivities, demonstrating the accuracy of the high-order method.

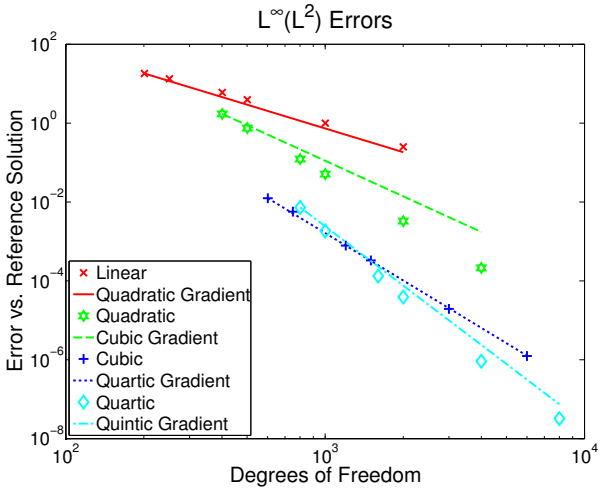
5.4.2 2D Homogeneous Conductivity

Having demonstrated the superior accuracy in 1D, we move on to 2D. In this subsection, unless otherwise noted we use a time-step $\Delta t = 0.01$ ms.

Before performing our experiments, we demonstrate the effect of anisotropy in our



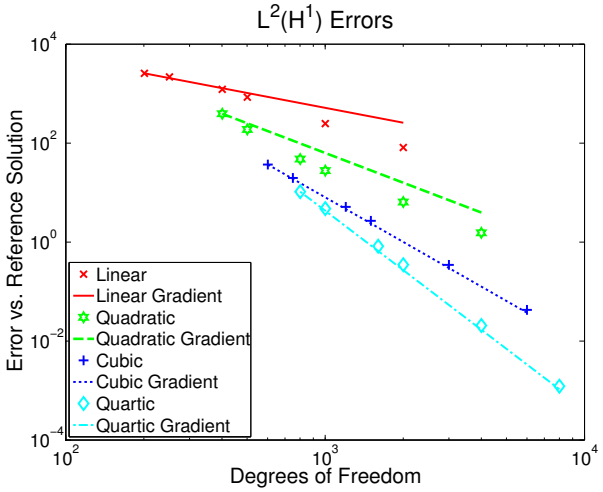
(a) Standard LR91 cell model



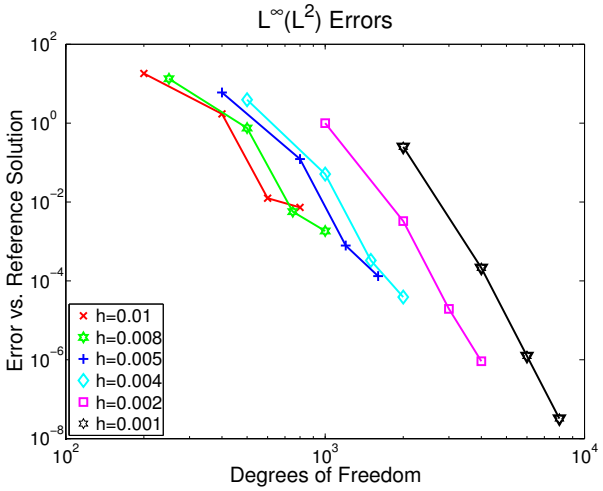
(b) Noble-form LR91 cell model

Figure 5.5: Data from 20 ms simulations on a 2 cm 1D domain using our method with and without the Noble-form LR91 cell model modification. Note the limited accuracy caused by the discontinuity in standard LR91. Δt=0.001 ms. $L^\infty(L^2)$ is the maximum-in-time of the L^2-norm of the error in space. The errors are against a quartic reference solution with $h = 10^{-4}$ cm.

simulations. Figure 5.8 shows wavefront locations (u=0 mV) with p=1-4 for two simulation modes at 12 ms, one with homogeneous isotropic conductivity (Figure 5.8(a)) and one with homogeneous anisotropic conductivity using fibres aligned to the x-axis (Figure 5.8(b)). In the latter case, the conductivity along the fibres is the same as that for the isotropic case (1 S cm^{-1}), and perpendicular to the fibres we use one-fifth of that value. Both simulations were initiated with the same stimulus current in the lower-left corner



(a) Sobolev norm errors



(b) Exponential Convergence in $L^\infty\left(L^2\right)$

Figure 5.6: Noble-form LR91 with our method. Figure 5.6(a) shows the $L^2(H^1)$ norm of the error; this is the L^2-in-time norm of the Sobolev $\|\cdot\|_{1,2}$ norm of the error in space. Figure 5.6(b) shows the same data as Figure 5.5(b), but with the exponential convergence in p highlighted instead. Data from 20 ms simulations on a 2 cm 1D domain. Δt=0.001 ms. The errors are against a quartic reference solution with $h = 10^{-4}$ cm. The values of h given are in cm.

of the domain and use a mesh with mean element diameter 0.0113 cm. Note the poor accuracy of the linear case, and note further that where the anisotropic $p = 1$ wavefront has propagated predominantly perpendicular to the fibres in Figure 5.8(b), its accuracy is even worse. This is due to the low conduction velocity in that direction, requiring better approximation properties to properly capture u at the wavefront. The higher-order sim-

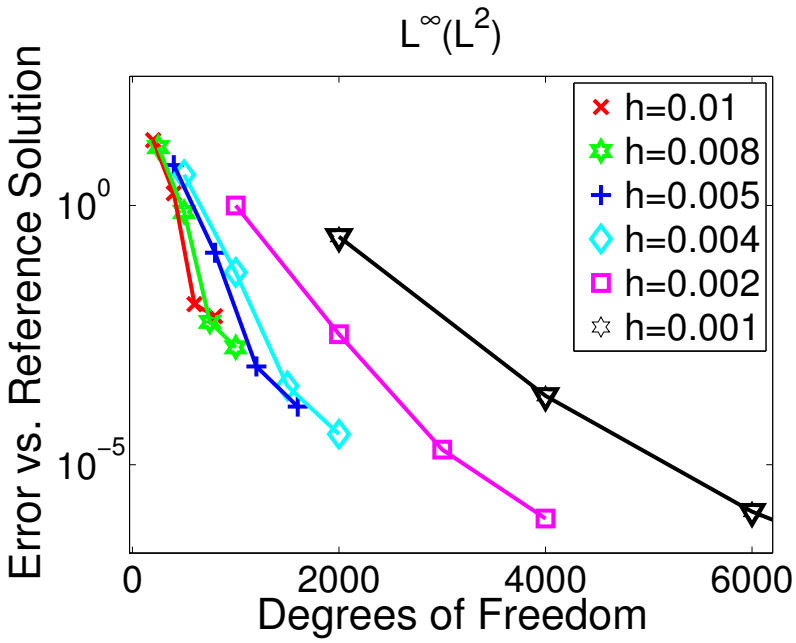


Figure 5.7: Noble-form LR91 with our method. Same data as in figure 5.6(b), but plotted on semilog axes to make the exponential convergence clearer. Data from 20 ms simulations on a 2 cm 1D domain. Δt=0.001 ms. The errors are against a quartic reference solution with $h = 10^{-4}$ cm. The values of h given are in cm.

ulations depicted do use considerably more degrees of freedom than the $p = 1$ simulation to which we compare them, and thus we could expect the improved accuracy in advance. However, because simulation studies in the literature generally do not use meshes finer than this, Figure 5.8 serves to expose the error that remains at such standard discretisation resolutions, both in terms of wavefront location (Figure 5.8(a)) and shape (Figure 5.8(b)).

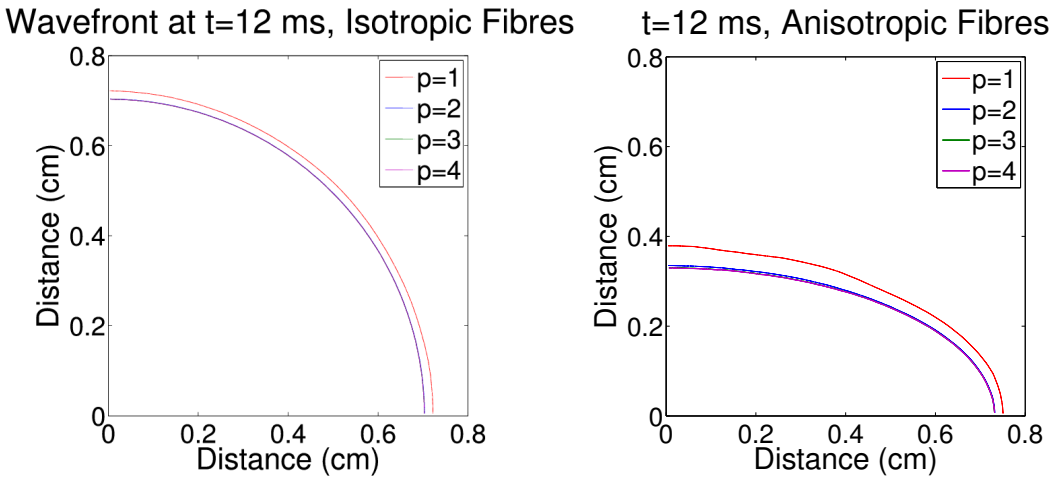
In our first 2D experiment we investigated a 1 cm by 1 cm 2D domain with homogeneous isotropic conductivity. A ramp stimulus was applied to a rectangular region along the bottom of the domain, generating a planar propagating wave. The measured CVs and activation times at the top-right corner of the domain are given in Table 5.2. Each simulation was performed with Δt=0.01 ms and Δt=0.001 ms (asterisked) in order to investigate how the temporal error affects the results. The CV is an average taken over many randomly selected point pairs in the domain. Percentage errors in conduction ve-

		Conductivity=1 Sm^{-1}			Conductivity=0.2 Sm^{-1}		
Elt Diam h (cm)	p	Activation Time (ms)	CV $(cm\ s^{-1})$	CV % Err	Activation Time (ms)	CV $(cm\ s^{-1})$	CV % Err
0.1	1	50.43	118.13	83.09	62.76	94.65	228.30
	2	67.01	88.50	37.17	103.18	57.18	98.34
	3	80.94	73.13	13.34	182.44	32.09	11.31
	4	88.62	66.73	3.43	243.44	24.11	16.37
0.05	1	68.33	86.81	34.55	104.77	56.34	95.42
	2	81.35	72.81	12.85	143.15	41.17	42.80
	3	88.68	66.73	3.43	177.89	33.10	14.81
	4	90.39	65.47	1.47	198.47	29.65	2.84
0.02	1	83.92	70.55	9.34	157.40	37.44	29.86
	2	90.61	65.30	1.21	185.18	31.81	10.34
	3	91.65	64.54	0.03	199.50	29.52	2.39
	4	91.66	64.54	0.03	202.17	29.12	1.01
0.01	1	89.36	66.20	2.60	183.75	32.06	11.20
	2	91.59	64.58	0.09	200.79	29.33	1.73
	3	91.67	64.52	0.00	204.10	28.84	0.03
	4	91.67	64.52	0.00	204.16	28.84	0.03
0.005	1	91.08	64.96	0.68	197.83	29.77	3.26
	2	91.67	64.54	0.03	203.93	28.87	0.14
	3	91.67	64.52	0.00	204.22	28.83	0.00
	4	91.67	64.52	0.00	204.22	28.83	0.00
0.002	1	91.58	64.60	0.12	203.15	28.98	0.52
	2	91.67	64.52	0.00	204.21	28.83	0.00
	3	91.67	64.52	0.00	204.22	28.83	0.00
	4	91.67	64.52	0.00	204.22	28.83	0.00
0.001	1	91.65	64.54	0.03	203.95	28.86	0.10
	2	91.67	64.52	0.00	204.22	28.83	0.00
	3	91.67	64.52	0.00	204.22	28.83	0.00
	4	91.67	64.52	0 (def)	204.22	28.83	0 (def)

Table 5.1: 6 cm 1D domain simulation with a ramp stimulus at one end and $\Delta t = 0.01$ ms. The activation time is the time until the transmembrane potential at the node at the opposite end of the domain (at 6 cm) passes up through 0 mV. CVs are computed using the nodes at 2 cm and 4 cm. Highlighted in green are the coarsest mesh results for a CV error below 5% in the $p = 1$ and $p > 1$ cases, and in blue for CV error below 1%.

locity are also presented in the table, using the finest quartic simulation with the same value of Δt as a reference. Note the high accuracy of the quartic simulations, and the minimal variation in the percentage errors caused by varying Δt. Details on the degrees of freedom are in Table 5.5.

Highlighted in blue, the data show that p=4 on the coarsest mesh or p=3 on the second coarsest mesh with Δt=0.01 ms ought to be preferred over p=1 on the finest mesh, given that this produces given that these produce, respectively, 2.5 times less error with a linear system solve which takes two-thirds of the time to solve, and a sixfold-



(a) Isotropic (b) Anisotropic

Figure 5.8: Wavefront location at $t = 12$ ms with p=1-4, demonstrating the scheme with isotropic tissue and with fibres aligned with the x-axis having anisotropic conductivity (1 S m^{-1} along the fibres and 0.2 S m^{-1} perpendicular to them). In both cases, the larger p is, the less distance the wavefront has propagated. Note that $p = 2 - 4$ are indistinguishable in Figure 5.8(a), and that $p = 3$ and $p = 4$ are indistinguishable in Figure 5.8(b). This simulation used Δt=0.01 ms, the mean element diameter was 0.0113 cm and the stimulus was applied to the lower-left corner of the domain.

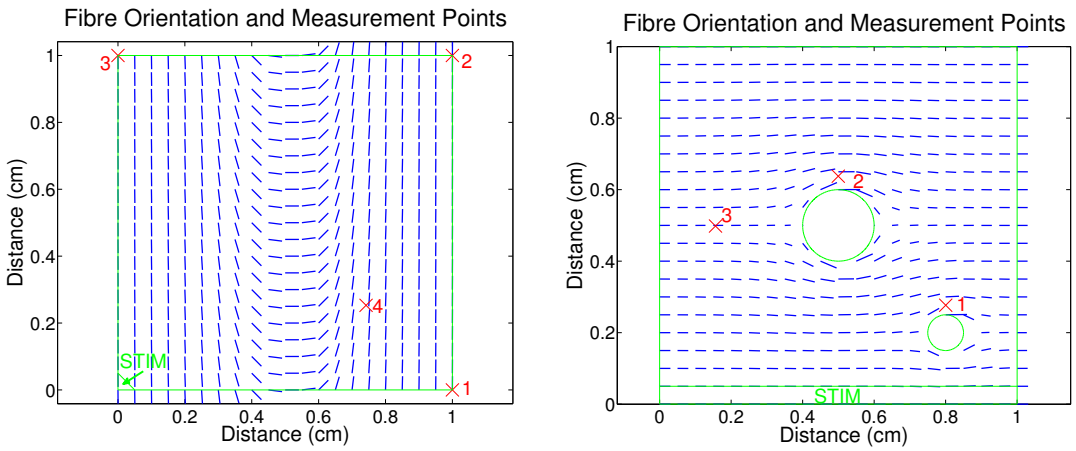
smaller error with a faster linear system. Alternatively, p=2 on the second coarsest mesh with Δt=0.01 ms produces roughly equivalent accuracy to the finest linear solution, but does so using a linear system which can be solved four times faster. Similar results hold for the $\Delta t = 0.001$ ms case, highlighted in green.

5.4.3 2D Inhomogeneous Conductivity: Plain Square Domain with Cubic Fibre Field

We studied simulation on a 1 cm by 1 cm domain using $\Delta t = 0.01$ ms and inhomogeneous anisotropic conductivity, with the conductivity along the fibres the same as that used in the isotropic case above (1 Sm^{-1}), and one-fifth of this value in the perpendicular direction. The fibre field is defined by a cubic polynomial designed to represent cardiac fibres rapidly changing orientation (see Figure 5.9(a)) and is integrated by reading the value of the vector field at each Gauss point [118] during matrix construction. The domain

Mean Element Diameter h (cm)	Basis Degree p	Activation Time (ms)	CV ($cm\ s^{-1}$)	% CV Error vs. best solution	ILU PCG Time (s)	Mean It. Time (s)
0.0444	1	12.89	81.93	26.37	0.002	0.050
	2	14.89	69.02	6.44	0.008	0.056
	3	15.41	66.28	2.23	0.020	0.067
	4	15.63	65.00	0.24	0.061	0.109
	1*	12.72	83.01	26.85	0.002	0.051
	2*	14.72	69.84	6.72	0.007	0.056
	3*	15.25	66.99	2.37	0.015	0.063
	4*	15.49	65.58	0.21	0.050	0.097
0.0222	1	14.61	70.59	8.86	0.005	0.190
	2	15.55	65.41	0.88	0.023	0.210
	3	15.64	64.91	0.11	0.070	0.257
	4	15.66	64.84	0.00	0.179	0.368
	1*	14.43	71.44	9.16	0.005	0.198
	2*	15.39	66.06	0.95	0.022	0.209
	3*	15.49	65.51	0.11	0.059	0.249
	4*	15.51	65.44	0.00	0.170	0.361
0.0111	1	15.63	66.40	2.41	0.015	0.854
	2	15.65	64.89	0.08	0.074	0.912
	3	15.65	64.85	0.01	0.226	1.056
	4	15.65	64.84	0.01	0.592	1.423
	1*	15.19	67.08	2.50	0.016	0.865
	2*	15.49	65.50	0.09	0.084	0.334
	3*	15.50	65.44	0.00	0.236	1.125
	4*	15.50	65.44	0.00	0.663	1.530
0.0055	1	15.58	65.24	0.61	0.087	3.585
	2	15.65	64.85	0.01	0.423	4.027
	3	15.65	64.84	0.00	1.432	4.972
	4	15.65	64.84	0 (by def.)	3.841	7.397
	1*	15.42	65.86	0.65	0.065	3.681
	2*	15.50	65.45	0.02	0.342	4.100
	3*	15.50	65.44	0.00	0.984	4.655
	4*	15.50	65.44	0 (by def.)	2.462	6.135

Table 5.2: Activation times, conduction velocities, percentage conduction velocity errors and linear system solve times for a variety of 2D meshes of $\Omega = [0, 1] \times [0, 1]$ cm. The first timings presented are for a single time-step only and use preconditioned conjugate gradients (PCG) with an incomplete LU (ILU) decomposition of the portion of the system matrix corresponding to the linear basis functions as a preconditioner, and the second times all code on a time-step iteration (ILU PCG + cell updates, etc.). The stimulus was along the bottom edge of the domain and the activation time is for the node in the top-right corner. Asterisked basis degrees indicate that the simulation was run with $\Delta t = 0.001$ ms instead of the usual $\Delta t = 0.01$ ms. The percentage errors use as a reference the p=4 simulation on the finest mesh with the same value of Δt. Some interesting results are highlighted; see the text for details.



(a) Plain square domain cubic fibre field.

(b) Domain with holes.

Figure 5.9: Fibre orientations used for the inhomogeneous simulations. The marked points are those at which activation time is measured. For Figure 5.9(a), results are displayed in Table 5.3 and stimulation was a ramp in the bottom-left corner. For Figure 5.9(b), results are displayed in in Table 5.4 and stimulation was a ramp along the bottom of the domain.

was stimulated with a ramp in the bottom-left corner of the domain. The measured activation times at the four points shown in Figure 5.9(a) are given in Table 5.3. Note how cubic and quartic elements display superior accuracy to all tested linear simulations by the second coarsest level, and that equivalent accuracy to the finest $p = 1$ solution can be obtained on the second coarsest mesh using $p = 2$ (blue highlighting). Notice also how large the $p = 1$ errors are, highlighted in green. Details on the degrees of freedom are in Table 5.5.

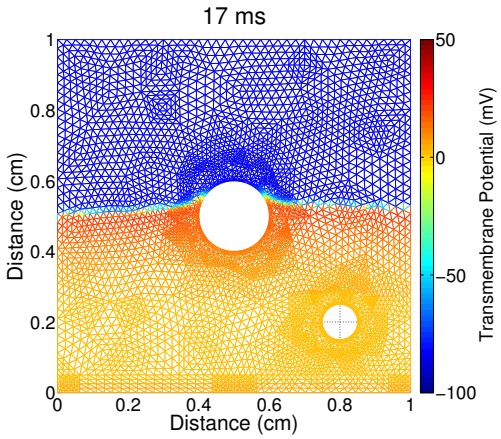
5.4.4 2D Inhomogeneous Conductivity: Domain with Holes

Figure 5.9(b) shows the fibre vector field on a 1 cm by 1 cm domain with two holes representing blood vessels passing through the simulation plane. We performed this study to demonstrate the applicability of the method when the domain contains fine structure that must be properly captured using small elements. This vector field was generated using the Laplace-Dirichlet approach and the time-step used for the monodomain was $\Delta t = 0.01$ ms. The activation times at the three points shown in Figure 5.9(b) are given

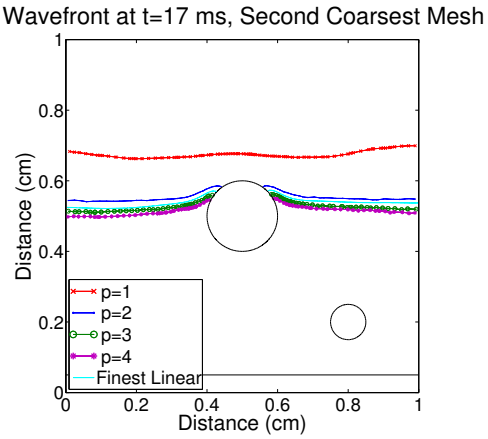
Mean Elt Diam h (cm)	p	Activation Time 1 (ms)	Activation Time 2 (ms)	Activation Time 3 (ms)	Activation Time 4 (ms)	Max % Err
0.0452	1	18.29	22.64	13.17	15.68	40.63
	2	24.55	29.03	15.12	20.51	20.32
	3	27.89	31.86	15.66	22.88	9.48
	4	35.86	37.63	16.04	28.65	16.39
0.0226	1	23.60	28.07	14.90	19.77	23.40
	2	28.52	32.74	15.93	23.42	7.43
	3	30.01	33.95	16.06	24.48	2.60
	4	30.72	34.46	16.09	24.99	0.29
0.0113	1	27.52	31.77	15.72	22.71	10.52
	2	30.38	34.22	16.08	24.74	1.40
	3	30.75	34.47	16.09	24.98	0.19
	4	30.80	34.50	16.09	25.02	0.03
0.0057	1	29.68	33.61	15.99	24.25	6.91
	2	30.77	34.48	16.09	24.99	0.13
	3	30.81	34.50	16.09	25.02	0.00
	4	30.81	34.50	16.09	25.02	0 (def)

Table 5.3: Activation times in the 1 cm by 1 cm domain with fibre orientation and four measurement points shown in Figure 5.9(a) using various h and p values. The maximum CV error over the four points is shown, using the finest quartic solution as a reference.

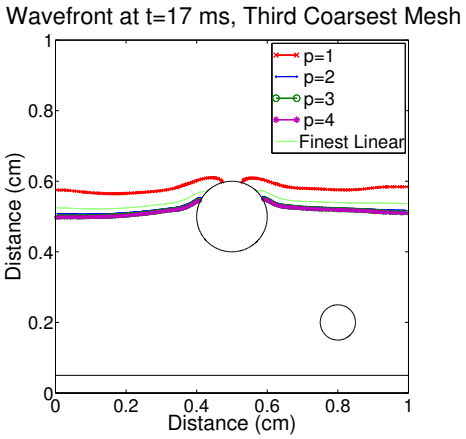
in Table 5.4. With blue highlighting, our results show that the third coarsest mesh with $p = 2$ can more than halve the worst percentage error in activation time using a linear system which can be solved in around the same time as for $p = 1$ on the finest mesh, or with $p = 4$ on the second coarsest mesh, we can reduce the error to one-tenth that of p=1 on the finest mesh without having to resort to extremely fine meshes. See Figure 5.10 for some wavefront locations using various meshes and values of p; note how poor the wavefront location can be when $p = 1$ even on highly refined meshes, and how on the second coarsest mesh with $p = 3$ the results are better than $p = 1$ on the finest mesh. As with Figure 5.8, we naturally expect the observed improvement in accuracy due to the increasing number of degrees of freedom as p increases on a fixed mesh. However, Figure 5.10 provides a visual understanding of how accuracy increases with p, and in particular Figure 5.10(d) demonstrates the magnitude of the error that remains in $p = 1$ simulation results when the conductivity is low, and when using a value of h not dissimilar to those found in the simulation literature.



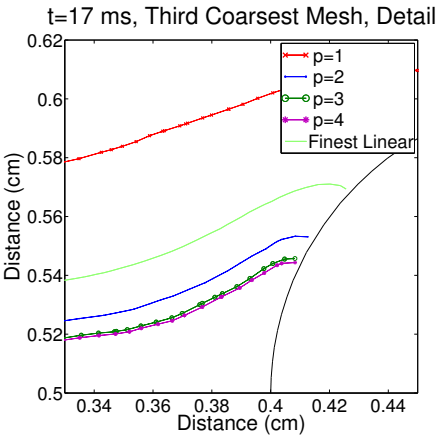
(a) Two holes mesh, second coarsest, p=4.



(b) Two holes mesh, second coarsest, p=1-4 and finest mesh p=1.



(c) Two holes mesh, third coarsest, p=1-4 and finest mesh p=1.



(d) Detail from Figure 5.10(c).

Figure 5.10: Wavefront location at 17 ms using various meshes and degrees. Figure 5.10(a): transmembrane potential distribution on the second coarsest mesh with p=4. Note that each triangle represents a single element of the mesh, and that this is independent of p. 5.10(b): Wavefront ($u = 0$) contours on the second coarsest mesh with various degrees compared to p=1 on the finest mesh. 5.10(c): Wavefront contours on the second coarsest mesh with various degrees compared to p=1 on the finest mesh. 5.10(d): Detail from Figure 5.10(c). See the corresponding activation time data in Table 5.4.

5.4.5 2D Degrees of Freedom

Details on the degrees of freedom for the various meshes and basis degrees are presented in Table 5.5.

Max. Elt Diam h (cm)	p	Activation Time 1 (ms)	Activation Time 2 (ms)	Activation Time 3 (ms)	Max % Error	ILU PCG Time (s)	Mean It. Time (s)
0.0593	1	6.46	13.48	10.01	42.31	0.003	0.109
	2	7.61	17.16	13.69	21.10	0.017	0.121
	3	7.96	18.68	15.59	10.14	0.055	0.160
	4	8.15	20.66	18.65	7.49	0.152	0.257
0.0296	1	7.24	16.14	13.08	24.61	0.007	0.304
	2	8.00	18.84	15.96	8.01	0.039	0.321
	3	8.12	19.54	16.86	2.82	0.122	0.402
	4	8.14	19.85	17.29	0.35	0.323	0.607
0.0148	1	7.71	18.11	15.36	11.47	0.020	1.029
	2	8.11	19.68	17.08	1.56	0.118	1.142
	3	8.14	19.86	17.31	0.23	0.396	1.142
	4	8.15	19.89	17.35	0.00	1.225	2.322
0.0076	1	7.96	19.21	16.64	4.09	0.102	4.224
	2	8.14	19.86	17.32	0.18	0.878	4.700
	3	8.15	19.89	17.35	0.00	4.969	8.981
	4	8.15	19.89	17.35	0 (def)	15.125	19.086

Table 5.4: Activation times in the 1 $cm \times 1$ cm domain with holes, fibre orientation and three measurement points shown in Figure 5.9(b) using various h and p values. The meshes are identified by maximum element diameter because the mean element size would convey little information due to the very small elements near the holes. The timings presented are for the linear system only (ILU PCG) and for the whole-timestep (ILU PCG + cell updates, etc.). More information about the meshes used is presented in Table 5.5, and some plots of the wavefront at 17 ms are shown in Figure 5.10. Results showing the accuracy improvement over $p = 1$ are highlighted in blue; in green we highlight how large the $p = 1$ errors are even on quite fine meshes.

		Degrees of Freedom With:		
Mesh Refinement Level	Basis Degree p	Homogeneous Conductivity	Inhomogeneous Conductivity	Two Holes
0	1	786	733	1,770
	2	3,047	2,841	6,764
	3	6,784	6,325	14,981
	4	11,997	11,185	26,421
1	1	3,047	2,841	4,500
	2	11,997	11,185	17,596
	3	26,851	25,033	39,287
	4	47,609	44,385	69,573
2	1	11,997	11,185	14,482
	2	47,609	44,385	57,352
	3	106,837	99,601	128,609
	4	189,681	176,833	228,253
3	1	47,609	44,385	52,199
	2	189,681	176,833	207,854
	3	426,217	397,345	466,964
	4	757,217	705,921	829,529

Table 5.5: The degrees of freedom for the 2D simulations.

5.5 Uniform Degree High-Order Simulations in Three Dimensions

In three dimensions, we work with an additive Schwarz preconditioner, which has been suggested in the literature as the best class of preconditioner to use for this problem[1] [102]. Because applying this preconditioner involves solving lots of small, local linear systems, it is very easily parallelisable and thus well-suited for simulation using supercomputing facilities. However, when all these small linear systems have to be solved in serial, as in our software, solving the global linear system becomes slower than it would be if we were to use a simpler preconditioner. Therefore, the timings that we produce are only meaningful when used to make comparisons between simulations which use an additive Schwarz preconditioner. For this reason, all our 3D simulations use this preconditioner, which is described in Section 4.6.3. Timings aside, we shall see that the preconditioner also performs well in terms of the number of iterations that the conjugate gradients algorithm requires to converge. We use a tolerance of 10^{-6} for our Schwarz-preconditioned conjugate gradient algorithm; this was tested to ensure that it gives the same activation times as a tolerance of 10^{-10}.

As for Section 5.4, all simulations were performed in MATLAB and use a semi-implicit backward Euler time discretisation scheme, we use a fixed $\tilde{p} = 4$ regardless of the value of $p(\leq 4)$ and timings presented are for a 3.4 GHz CPU coupled where necessary with an AMD Radeon 6950 graphics card. Note that in three dimensions we use the ten Tusscher 2006 cell model [57], as opposed to the simpler Luo-Rudy I model of the one- and two-dimensional simulations [34].

[1] Specifically, a hybrid multi-level Schwarz preconditioner is best, but we work with a simpler form here for which we also obtain excellent PCG iteration counts.

5.5.1 Small Test Cube

Because of the time-consuming nature of performing monodomain simulation in three dimensions, a thorough evaluation of a numerical method across a range of different parameters is a demanding task. Since the software we are working with here is non-parallel, three-dimensional testing is all the more difficult. For this reason, we begin by performing an evaluation of high-order finite elements on a very small test domain: a cube of myocardial tissue having dimensions $0.2 \times 0.2 \times 0.2$ cm. We set anisotropic conductivities equal to those used for the Niederer benchmark problem, given by 1.33 S m^{-1} parallel to the fibres and 0.18 S m^{-1} in their orthogonal plane, with the fibres aligned with one of the coordinate axes [58]. We record the activation time of the last node in the domain, in the corner of the cube furthest from the stimulus site; this is labelled as node eight in Figure 5.11.

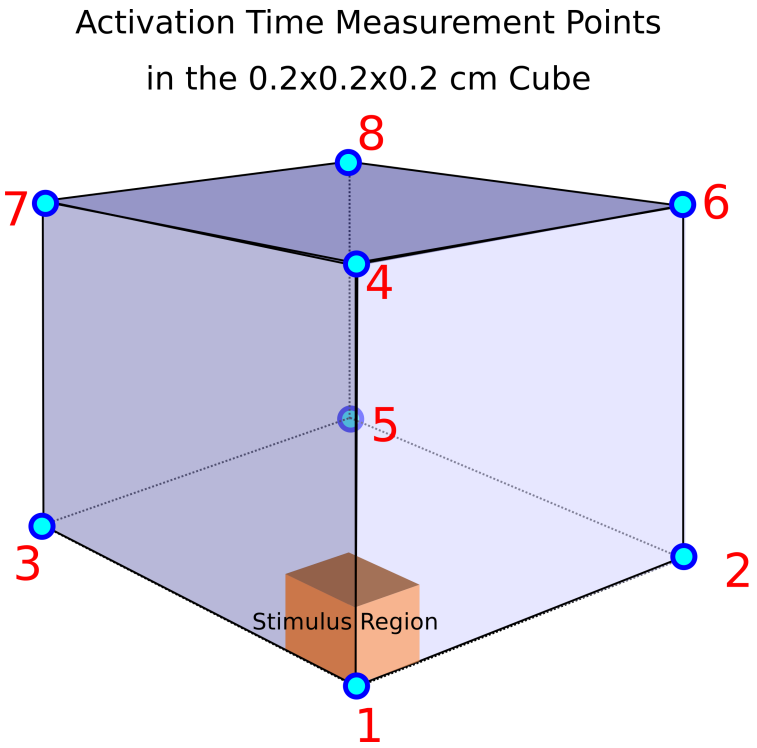


Figure 5.11: The small test cube with the stimulus region and activation time measurement points marked. We generally only report activation times at node 8 (referred to as the "last node"), but use all nodes in Section 7.2.2.

We present the results of these small cube mesh simulations in Table 5.6. Notice that the percentage activation time errors (using a h=0.01 cm p=4 simulation as a reference solution) remain quite large even on the finest mesh. This large error is in part due to the lower conductivities that we use; as these are physiological it is important to take them into account when evaluating a numerical method. Our higher order scheme does a much better job of cheaply producing very accurate solutions, for example by preferring the $p = 3$ coarsest mesh solution over the one generated with $p = 1$ on the finest mesh, we can produce a solution with one-third of the error in about one-third of the time. If we instead chose the $p = 4$ coarsest mesh solution, we would get a solution with a thirty-three times lower error in approximately the same amount of CPU time. It is also important to note that the Schwarz preconditioner we are using means that the PCG iteration counts remain small, and in a manner which appears to be independent of h and only weakly dependent on p for $p \neq 1$. We will return to this problem in Section 6.7.1 in order to evaluate the performance of spatial adaptivity.

Elt. Diam. h (cm)	Simulation Mode	Final Activ. Time (ms)	% Activ. Time Error	PCG Time (s)	Mean It. Time (s)	Mean PCG Iterations	Max. PCG Iterations
0.04	p=1	8.32	31.4	0.05	0.15	7.3	9
	p=2	8.93	26.4	0.12	0.22	12.3	16
	p=3	12.77	5.3	0.39	0.49	13.3	16
	p=4	12.19	0.5	1.57	1.68	14.3	17
0.02	p=1	8.58	29.3	0.23	0.52	6.7	9
	p=2	10.68	12.0	0.76	1.06	12.1	14
	p=3	12.18	0.4	2.90	3.21	12.4	15
	p=4	12.07	0.5	12.61	12.93	13.4	16
0.01	p=1	10.11	16.7	1.39	3.19	6.0	8

Table 5.6: Activation times of the final node in the $0.2 \times 0.2 \times 0.2$ cm small cube domain shown in Figure 5.11 using various h and p values. The reference solution, generated using h=0.01 and p=4, gave an activation time of 12.13 ms. The simulation mode is given in terms of p, for the fixed polynomial degree of the simulation. Fibres and conductivities are as given in [58]. The timings presented are for the linear system only (additive Schwarz PCG) and for the whole-timestep (additive Schwarz PCG + cell updates, etc.). The mean number of iterations that the PCG took to converge are given. More information about the meshes used is presented in Table 5.8.

5.5.2 The Niederer Benchmark

We performed simulation in three dimensions using the benchmark designed by Niederer et al. [58]. This is a reasonably well-defined problem intended to test the accuracy of cardiac electrophysiology software. The simulation domain is given to be $0.3 \times 0.7 \times 2$ cm at mesh discretisations of 0.05, 0.02 and 0.01 cm, uses fibres aligned to the long axis of the domain and conductivities set to 1.33 S m^{-1} parallel to the fibres and 0.17 S m^{-1} perpendicular to them. The stimulus is applied to the cubic region $[0, 0.15]^3$ cm^3, and the activation times evaluated at the points shown in Figure 5.12. Recall that in 3D we use the ten Tusscher 2006 cell model.

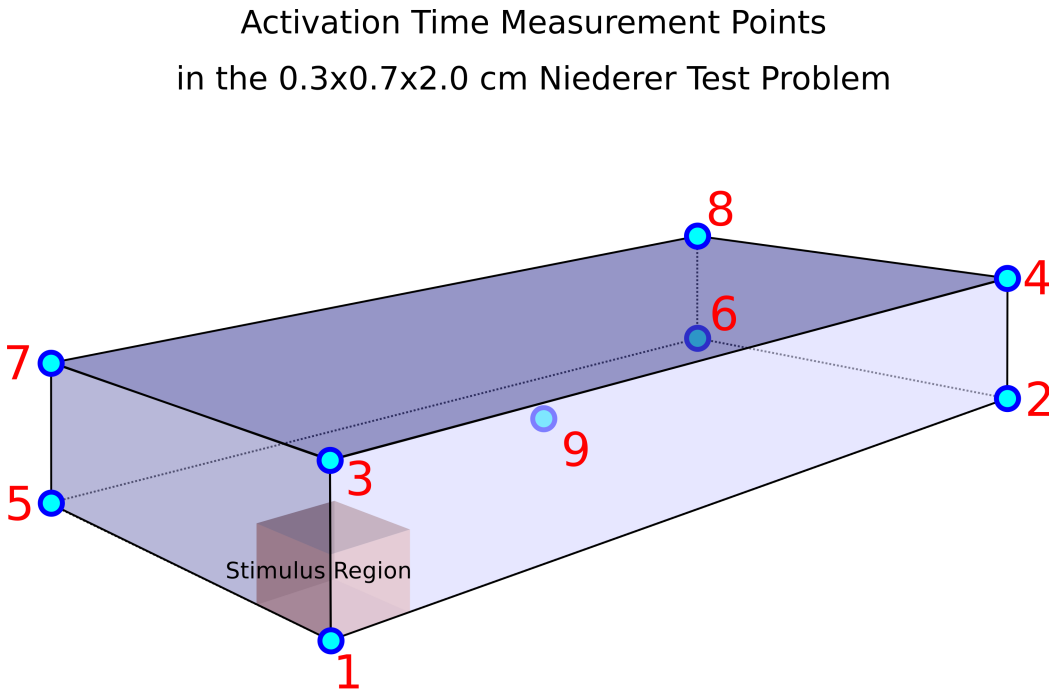


Figure 5.12: The Niederer benchmark domain with the stimulus region and activation time measurement points marked. Note that the ninth node is at the centre of mass of the domain.

The results of solving this test problem with our software are presented in Table 5.7. Notice that the activation times are more accurate at almost all of the nodes when we use $p = 3$ or 4 on the coarsest mesh than with $p = 1$ on the finest, and that in those cases, the simulation time is 20-80% of that for the finest $p = 1$ simulation. We evaluate

this problem further with spatial adaptivity in Section 6.7.2.

h	p	Activation Time in ms at Node:									PCG Time (s)	Mean It. Time (s)
		1	2	3	4	5	6	7	8	9		
0.05	1	1.25	27.95	5.87	27.12	15.96	29.24	14.40	33.22	13.40	0.63	1.48
	2	1.24	30.08	6.29	30.03	19.45	33.49	19.15	32.70	15.86	2.50	3.33
	3	1.24	31.52	7.78	32.85	23.10	38.24	26.33	40.80	18.66	9.54	10.37
	4	1.24	31.90	7.54	33.66	25.21	40.02	28.89	41.52	19.24	42.85	43.68
0.02	1	1.23	30.64	7.12	30.47	21.65	33.56	21.15	32.73	15.89	-	-
	2	1.24	31.92	7.81	32.69	24.30	39.16	25.11	39.47	18.61	-	-
	3	1.24	32.19	8.46	33.41	26.65	41.84	28.97	43.01	19.93	-	-
	4	1.25	32.23	8.37	33.52	26.64	42.16	28.78	43.20	20.06	-	-
0.01	1*	1.23	31.46	7.82	31.93	25.44	38.04	26.02	38.17	17.87	51.58	51.82

Table 5.7: 3D activation times on the $0.3 \times 0.7 \times 2$ cm Niederer test problem domain shown in Figure 5.12 using various h (cm) and p values. The timings presented are for the additive Schwarz PCG-solve only, and for the whole-timestep full iteration (PCG, cell updates, etc.). Plots of the wavefront at 10 ms are shown in Figure 5.13. The asterisked simulation was run with $\tilde{p} = 1$, and some timings are absent because the simulation was run on a different computer, due to memory restrictions on the primary test machine.

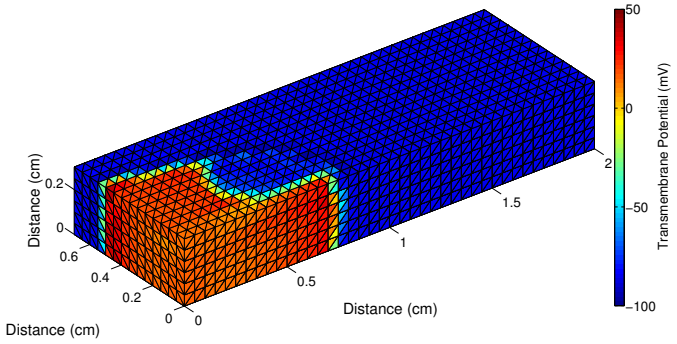
Figure 5.13 shows the wavefront at 10 ms on the h=0.05 cm mesh with p=1-4, showing the convergence in p visually and demonstrating the difference that high-order FEM makes. Notice that the inaccuracy that occurs due to p=1 FEM with a coarse mesh affects more than just activation times; we can see from these figures that the shape of the wave is substantially altered. This is because the CV error is larger in directions perpendicular to the fibres due to the lower conductivity.

5.5.3 3D Degrees of Freedom

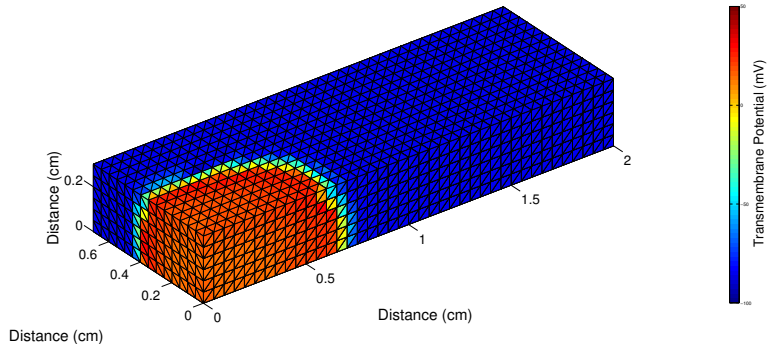
Details on the degrees of freedom for the various meshes and basis degrees are presented in Table 5.8.

		Degrees of Freedom For:	
Element Diameter (cm)	Basis Degree p	Small Cube Domain	Niederer Benchmark Cuboid
0.04 (cube) / 0.05 (Niederer)	1	216	4,305
	2	1,331	30,537
	3	4,096	98,857
	4	9,261	229,425
0.02	1	1,331	58,176
	2	9,261	442,401
	3	29,791	1,467,676
	4	68,921	3,449,001
0.01	1	9,261	442,401
	2	68,921	-
	3	226,981	-
	4	531,441	-

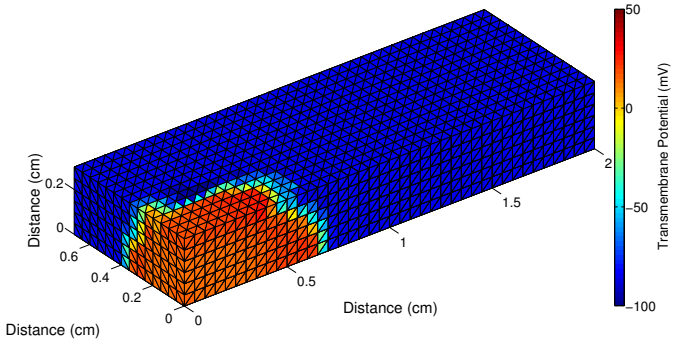
Table 5.8: The degrees of freedom for the 3D simulations. The empty entries are for simulations that were not performed due to hardware constraints.



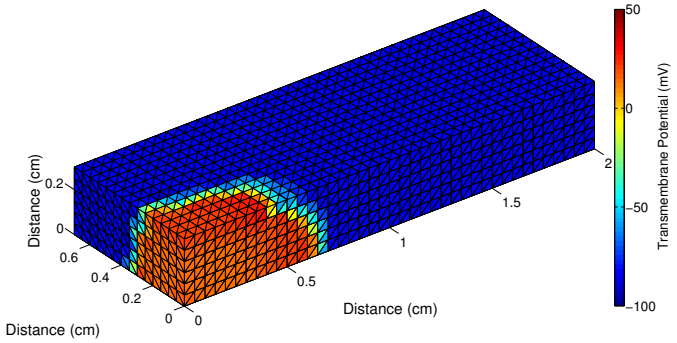
(a) p=1



(b) p=2



(c) p=3



(d) p=4

Figure 5.13: The state of the domain after 10 ms of simulation on the h=0.05 cm Niederer mesh with p=1-4.

Chapter 6

Adaptive High-Order Finite Elements for the Monodomain System

So far we have seen how the uniform high-order FEM is applied to the monodomain system and we have understood that the cell model must be treated as a PDE and discretised to the same high degree as the transmembrane potential. We know that we can achieve improved efficiency with this method, but because we have chosen to use hierarchical basis functions, we can do even better. In this chapter, we shall see that these allow us to create an adaptive version of the scheme which, unlike other adaptive methods, does not suffer from large computational overhead at the matrix-adjustment stage.

We describe and analyse the error indicator that we use to drive the adaptivity, explain the preconditioning strategies, demonstrate the excellent error-control properties, and then investigate performance of the adaptive scheme using similar test problems to those in the previous chapter. Simulations using a whole-heart mesh are included, and we perform a calculation based on these which indicates possible simulation speed improvements for a high-performance computing environment.

The essence of the adaptive scheme which we develop in this chapter is that we generally do not work with the entire space S_{hp} described in Section 4.5; rather, we

cut down on the required computational effort by adaptively shaping our solution space $S = S_{h\boldsymbol{p}} \subseteq S_{hp}$ to meet the requirements of the current state of the system. Here, $\boldsymbol{p} = (p_1, p_2, \ldots, p_{N_e})$ is a vector with one entry for each element of the mesh, with p_i the polynomial degree automatically selected for element ϵ_i on the present time-step (as we shall see in Section 6.3). Note that we now fix the cell model degree $\tilde{p} = p \geq p_i$ for all i, and that this definition of $\boldsymbol{p}$ now supersedes that of the previous chapter, where it was used to mean the pair $(p, \tilde{p})$. We also now drop the subscripts h from our finite element approximations. The adaptive shaping of S allows us to focus the DOF only where they are needed, meaning that the DOF used are only those required in order to give us confidence of producing an approximation in which the error has been controlled by some choice of a target error parameter θ. Thus we can obtain a high degree of computational efficiency, which we shall show allows us to quickly obtain highly accurate approximations.

The property of the hierarchical basis noted in Section 4.5 that on each element the basis of degree p is the same as the basis of degree $p+1$ with the degree $p+1$ functions removed makes p-adaptivity a computationally feasible method; in order to adapt, we simply add or remove the basis functions to or from the existing ones as necessary on each element; this can be achieved cheaply using pre-computed data. Without this property, we would be forced to reconstruct the linear system whenever we adapted it; the cost of this would outweigh the savings made by using the adapted system.

6.1 *A Posteriori* Error Indicator

Because we are to use spatial p-adaptivity, we need to know how much contribution to the total error occurs in each element, so we require a spatially-localised error indicator. We give an outline of the analytically-derived indicator η which can be computed on each element that we use in what follows. System (5.1) has weak form: *find* $u \in L^2\left(0, T; \mathcal{H}^1(\Omega)\right)$

and $w \in L^2\left(0,T;\left(L^2(\Omega)\right)^m\right)$ *such that* $\forall\ t \in [0,T]$,

$$\begin{aligned}\beta C_m\left(u_t,\chi\right)_\Omega + \left(\sigma\nabla u,\nabla\chi\right)_\Omega &= \beta\left(I_{total},\chi\right)_\Omega \quad \forall\chi\in\mathcal{H}^1(\Omega),\\ \left(w_t,\hat{\chi}\right)-\left(g,\hat{\chi}\right) &= 0 \quad \forall\hat{\chi}\in\left(L^2(\Omega)\right)^m.\end{aligned}$$

The two forms that we will work with here are the semidiscrete in time formulation: *for* $n=1,\ 2,\ \ldots,N$, *find* $u^n\in\mathcal{H}^1(\Omega)$ *and* $w^n\in\left(L^2(\Omega)\right)^m$ *such that*

$$\begin{aligned}\beta C_m\left(u^n-u^{n-1},\chi\right)_\Omega + \Delta t\left(\sigma\nabla u^n,\nabla\chi\right)_\Omega &= \beta\Delta t\left(I^{n-1}_{total},\chi\right)_\Omega \quad \forall\chi\in\mathcal{H}^1(\Omega), \quad (6.1)\\ \left(w^n-w^{n-1},\hat{\chi}\right)_\Omega &= \Delta t\left(\hat{g},\hat{\chi}\right)_\Omega \quad \forall\hat{\chi}\in\left(L^2(\Omega)\right)^m,\end{aligned}$$

where $\hat{g}$ is $g(u,w)$ evaluated at u^{n-1} and a mixture of w^n and w^{n-1}, using the latter only when the former is not possible without requiring iterations to perform an update over a time-step and $I^{n-1}_{total} := I_{total}(u^{n-1},w^n)$, and the fully-discrete formulation: *for* $n=1,\ 2,\ \ldots,N$, *find* $u^n_{\boldsymbol{p}}\in S^n_{h\boldsymbol{p}}$ *and* $w^n\in\left(S_{h\tilde{p}}\right)^m$ *such that*

$$\begin{aligned}\beta C_m\left(u^n_{\boldsymbol{p}}-u^{n-1}_{\boldsymbol{p}},\chi_{\boldsymbol{p}}\right)_\Omega + \Delta t\left(\sigma\nabla u^n_{\boldsymbol{p}},\nabla\chi_{\boldsymbol{p}}\right)_\Omega &= \beta\Delta t\left(I^{n-1}_{total,\tilde{p}},\chi_{\boldsymbol{p}}\right)_\Omega \quad \forall\chi_{\boldsymbol{p}}\in S^n_{h\boldsymbol{p}}, \quad (6.2)\\ \left(w^n_{\tilde{p}}-w^{n-1}_{\tilde{p}},\chi_{\tilde{p}}\right)_\Omega &= \Delta t\left(\hat{g}_{\boldsymbol{p},\tilde{p}},\chi_{\tilde{p}}\right)_\Omega \quad \forall\chi_{\tilde{p}}\in\left(S_{h\tilde{p}}\right)^m,\end{aligned}$$

with $u^0_{\boldsymbol{p}}$ the L^2 projection of u^0 into $S^0_{h\boldsymbol{p}}$, $I^{n-1}_{total,\tilde{p}} := \Pi^{L2}_{\tilde{p}} I_{total}(u^{n-1}_{\boldsymbol{p}},w^n_{\tilde{p}})$, $\hat{g}_{\boldsymbol{p},\tilde{p}}$ is g evaluated at $u^{n-1}_{\boldsymbol{p}}$, $w^n_{\tilde{p}}$ and $w^{n-1}_{\tilde{p}}$, $\Pi^{L2}_{\tilde{p}}$ the L^2 projection into $S_{h\tilde{p}}$. Recall that w is an m-component vector. We require the bilinear form

$$E(v,w) := \beta C_m(v,w)_\Omega + \Delta t(\sigma\nabla v,\nabla w)_\Omega,$$

where $(\cdot,\cdot)_\Omega$ denotes the L^2 inner product on $L^2(\Omega)$.

Define the residual on element ϵ_i at time-step n by

$$r^i_n := \left(\beta C_m\frac{u^n_{\boldsymbol{p}}-u^{n-1}_{\boldsymbol{p}}}{\Delta t}-\nabla\cdot\left(\sigma\nabla u^n_{\boldsymbol{p}}\right)-\beta I^{n-1}_{total,\tilde{p}}\right)\Big|_{\epsilon_i},$$

where we have restricted the functions involved to the i-th element, and for an element ϵ_i and its neighbour $\epsilon_j \in \mathcal{M}$, with $\hat{n}_{\epsilon_i}$, $\hat{n}_{\epsilon_j}$ their outward-pointing unit normals along their shared edge (in 2D; face in 3D) $\gamma = \epsilon_i \cap \epsilon_j$ respectively, we write

$$\left[\frac{\partial u^n_{\boldsymbol{p}}}{\partial \hat{n}}\right]_\gamma := (\sigma \nabla u^n_{\boldsymbol{p}})_{\epsilon_i} \cdot \hat{n}_{\epsilon_i} + (\sigma \nabla u^n_{\boldsymbol{p}})_{\epsilon_j} \cdot \hat{n}_{\epsilon_j}$$

for the jump in the flux across γ. The *a posteriori* error indicator that we use to inform our adaptivity on each element ϵ_i after the n-th timestep is then given by

$$\eta_{\epsilon_i,n} = \Delta t \left(\|r^i_n\|^2_{0,2,\epsilon_i} \frac{h^2_{\epsilon_i}}{p^2_{\epsilon_i,n}} + \sum_{\gamma \in \partial \epsilon_i} \delta_\gamma \left\| \left[\frac{\partial u^n_{\boldsymbol{p}}}{\partial \hat{n}}\right]_\gamma \right\|^2_{0,2,\gamma} \frac{h_\gamma}{p_{\gamma,n}} \right)^{\frac{1}{2}}, \tag{6.3}$$

where $h_y := diam(y)$, and at time-step n, $p_{\epsilon_i,n} := polynomial\ degree(\epsilon_i)$ and $p_{\gamma,n} := min\{p_{\epsilon,n} | \gamma \subset \epsilon\}$. The term

$$\delta_\gamma = \begin{cases} 1, & \gamma \subset \partial\Omega \\ \frac{1}{2}, & \text{otherwise} \end{cases}$$

describes how we choose to distribute the jump-term error between two elements sharing an edge. The sum in Equation (6.3) is over the edges (2D, faces in 3D) which together make up the whole of the boundary $\partial\epsilon_i$ of ϵ_i. Similar error indicators have been used for elliptic problems previously; see for example [134], but the literature is lacking in their use for parabolic problems and for high-order finite elements, and certainly for cardiac applications.

We define the spatial discretisation error in the transmembrane potential to be the difference between the solutions to Equations (6.1) and (6.2) as $e^n_{\boldsymbol{p}} = u^n_{\boldsymbol{p}} - u^n$, and as we

will demonstrate, we have a bound on this of the form

$$\begin{aligned}\sqrt{\int_{t_{n-1}}^{t_n} E(e_{\boldsymbol{p}}^n, e_{\boldsymbol{p}}^n)dt} \;&\leq\; C\sum_{\epsilon_i\in\mathcal{M}}\eta_{\epsilon_i,n} \\ &+\; C\sqrt{\frac{\beta}{C_m}}\left[\Delta t^{\frac{3}{2}}\left(\sum_{j=1}^{m}\left\|w_{\tilde{p},j}^n - w_j^n\right\|_{0,2,\Omega} + \left\|\Pi_{\tilde{p}}^{L2} I_{total}^{n-1} - I_{total}^{n-1}\right\|_{0,2,\Omega}\right)\right. \\ &+\; \left.\left(\Delta t^{\frac{1}{2}}\sqrt{\beta C_m} + \Delta t^{\frac{3}{2}}\sqrt{\beta/C_m}\right)\left\|e_{\boldsymbol{p}}^{n-1}\right\|_{0,2,\Omega}\right], \qquad (6.4)\end{aligned}$$

where the first sum is over the mesh $\mathcal{M}$ and the second is over the m non-diffusing cell state variables, and $\Pi_{\tilde{p}}^{L2}$ is L^2 projection into the finite element space $S_{h\tilde{p}}$. This inequality shows how the error indicator η relates to the error over a single time-step.

In order to justify using η as an error indicator, we must suppose that the terms enclosed within the square brackets are small. For the term $\left\|\Pi_{\tilde{p}}^{L2} I_{total}^{n-1} - I_{total}^{n-1}\right\|_{0,2,\Omega}$, this is due to the fact that we use a large value of $\tilde{p}$; although we do not use the L^2 projection that the analysis calls for, in practice using a degree-$\tilde{p}$ interpolation projection $\Pi_{\tilde{p}}$, we suppose that our high order interpolation method is a good approximation to this. Evidence supporting this comes from the accuracy of the uniform scheme that we saw when using this same approximation in Chapter 5. The term $\sum_{j=1}^{m}\left\|w_j^n - w_{\tilde{p},j}^n\right\|_{0,2,\Omega}$ we have no error control for, but again we assume that the high value of $\tilde{p}$ that we use ensures this term is sufficiently small. The third term $\left\|e_{\boldsymbol{p}}^{n-1}\right\|_{0,2,\Omega}$ we presume has been kept small by the error control on the previous time-step. All of these assumptions hold sufficiently well to provide numerical results which behave as we would expect; see for example Figure 6.1 and Table 6.5.

Because Inequality (6.4) provides a bound which is dependent on the error that was present at the previous time-step, what it tells us is how much more error we introduce by taking a single step. We can derive an alternate, global bound which does not depend directly on the transmembrane potential error introduced on previous time-steps, and thus says what the error will look like by the N^{th} time-step. This time-global bound is given by

$$\begin{aligned}\|e_{\boldsymbol{p}}^N\|_{0,2,\Omega} &\leq \exp\left(\sum_{n=1}^{N-1}\frac{\Delta t}{2}\left(\frac{\beta C_m^2+L}{\beta C_m^2(1-\Delta t)}\right)\right)\left[\sqrt{\frac{\beta C_m^2+\Delta t L}{\beta C_m^2(1-\Delta t)}}\|e_{\boldsymbol{p}}^0\|_{0,2,\Omega}\right.\\ &+ \left.\sum_{n=1}^{N}\left(\sqrt{\frac{\Delta t D_n}{\beta C_m(1-\Delta t)}}+\sqrt{\frac{C\Delta t}{\lambda_{min}(\sigma)\beta C_m(1-\Delta t)}}\sum_{\epsilon_i\in\mathcal{M}}\eta_{\epsilon_i,n}\right)\right],\end{aligned}$$

where L is the Lipschitz constant for the current term I_{total}, see Inequality (6.13), $\lambda_{min}(\sigma)$ is the minimum eigenvalue of the conductivity tensor, and

$$D_n := \frac{1}{C_m}\left(L\sum_{j=1}^{m}\left\|w_j^n - w_{\tilde{p},j}^n\right\|_{0,2,\Omega}^2 + \left\|\Pi_{\tilde{p}}^{L2} I_{total}^{n-1} - I_{total}^{n-1}\right\|_{0,2,\Omega}^2\right)$$

measures the error in the cell model and in the projection of the current term. Note that this global bound requires that $\Delta t < 1$.

6.2 Deriving the Error Bound

In this section, we derive the error bounds which were given in Section 6.1 as a justification of the error indicator that we use.

6.2.1 Preliminaries

Suppose $\Omega \subset \mathbb{R}^2$ or $\mathbb{R}^3$ is an open domain with polygonal boundary $\partial\Omega$. Let $\mathcal{M}$ be a set meshing Ω, consisting of triangular or tetrahedral elements ϵ (see for example [135] or Section 4.1.2), let n be the index of the current time-step, let $S_{h\boldsymbol{p}}^n \subset \mathcal{H}^1(\Omega)$, $n = 1, 2, \ldots$ be a family of finite element spaces based on $\mathcal{M}$ consisting of piecewise polynomial functions of degree not greater than p, let $\mathcal{N}$ be the set of vertices of elements of $\mathcal{M}$ and let $\partial\mathcal{M}$ be the set of boundaries of elements of $\mathcal{M}$ consisting of triangle edges in 2D and tetrahedron faces in 3D. Suppose that for some $Q \geq 1$ as in Inequalities (4.6), the polynomial degrees $p_{i,n}$ (which define $S_{h\boldsymbol{p}}^n$) on the elements $\epsilon_i \in \mathcal{M}$ satisfies the quasiuniformity condition

$$Q^{-1}p_{j,n} \leq p_{i,n} \leq Qp_{j,n} \quad \forall\ \epsilon_i, \epsilon_j \in \mathcal{M} \text{ with } \bar{\epsilon_i} \cap \bar{\epsilon_j} \neq \emptyset \quad \forall n. \tag{6.5}$$

In what follows, C will be a generic constant whose value is independent of $\boldsymbol{p}$ but dependent on Q, and because its value is not relevant to this work, it will be varied without note. We use ideas from [136] and [137] as a guide.
We give the definitions

$$\begin{aligned}
\mathcal{N}(y) &:= \{V \in \mathcal{N} \mid V \in \bar{y}\} \text{ (the nodes of an object } y), \\
\omega_\gamma^1 &:= \cup_{V \in \mathcal{N}(\gamma)} \{\epsilon \in \mathcal{M} \mid V \in \bar{\epsilon}\} \text{ (the patch containing } \gamma \in \partial\mathcal{M}), \\
\omega_\epsilon^1 &:= \cup_{V \in \mathcal{N}(\epsilon)} \left\{\epsilon' \in \mathcal{M} \mid V \in \bar{\epsilon'}\right\} \text{ (the patch containing } \epsilon \in \mathcal{M}).
\end{aligned}$$

Then Theorem 3.1 of [138] implies that there exists a family of linear operators $\Pi_{h\boldsymbol{p}}^n : \mathcal{H}^1(\Omega) \to S_{h\boldsymbol{p}}^n$ such that for each n, $\epsilon_i \in \mathcal{M}$, $\gamma \in \partial\mathcal{M}$ and $w \in \mathcal{H}^1(\Omega)$,

$$\|w - \Pi_{h\boldsymbol{p}}^n w\|_{0,2,\epsilon_i} \leq C \frac{h_{\epsilon_i}}{p_{\epsilon_i,n}} \|\nabla w\|_{0,2,\omega_{\epsilon_i}^1}, \tag{6.6a}$$

$$\|w - \Pi_{h\boldsymbol{p}}^n w\|_{0,2,\gamma} \leq C \sqrt{\frac{h_\gamma}{p_{\gamma,n}}} \|\nabla w\|_{0,2,\omega_\gamma^1}. \tag{6.6b}$$

with C depending only on Q.
We define the element and edge inner products for regions $\mathcal{R}$ and boundary components γ by

$$(v, w)_{\mathcal{R}} = \int_{\mathcal{R}} vw \ dx, \qquad \langle v, w \rangle_\gamma = \int_\gamma vw \ ds,$$

respectively, where the multiplication in the expression for $(v, w)_{\mathcal{R}}$ is a dot product in the case where v and w are vectors.

6.2.2 A posteriori error estimation for the PDE

We will demonstrate that the use of η as an error indicator is justified by showing its role in bounding the error in the finite element approximation of u. We start from the PDE System 5.1, which we intend to solve using continuous hierarchical polynomial finite elements in space and a semi-implicit time-stepping method for solutions $u_{\boldsymbol{p}}^0, u_{\boldsymbol{p}}^1, \ldots, u_{\boldsymbol{p}}^N$ at times $0 = t_0, t_1, \ldots, t_N$. We write $\Delta t = t_n - t_{n-1}$ for the (constant) time-step.

Remark. *Recall that we actually work with the PDE system for the cell model w too, as given in System (5.1), but that we only treat error control for Equation (5.1a). In parallel, the time taken to solve for w is a tiny fraction of that required for the transmembrane potential u [139], and so the approach we have previously described of solving for w using Lagrange finite elements of fixed degree $\tilde{p}$ is both cheap and accurate, and if deemed necessary we can increase its accuracy cheaply. Thus, we do not concern ourselves with adapting it.*

Recall that u^n is the semidiscrete in time solution at time t_n, and $u_{\boldsymbol{p}}^n$ is its fully-discrete counterpart. Our aim is to identify a computable error indicator for controlling the error $e_{\boldsymbol{p}}^n = u_{\boldsymbol{p}}^n - u^n$ during simulations. Our initial approach is to bound $e_{\boldsymbol{p}}^n$ using the norm induced by the inner product

$$E(v, w) = \beta C_m (v, w)_\Omega + \Delta t (\sigma \nabla v, \nabla w)_\Omega$$

on $\mathcal{H}^1(\Omega)$. We define A_{n-1} as

$$A_{n-1} \quad := \quad \beta \| I_{total,\tilde{p}}^{n-1} - I_{total}^{n-1} \|_{0,2,\Omega}. \tag{6.7}$$

We prove the *a posteriori* error bound as follows.

Theorem 6.2.1. *Define the residual at the n-th time-step on ϵ_i by*

$$r_n^i := \beta C_m \frac{u_{\boldsymbol{p}}^n - u_{\boldsymbol{p}}^{n-1}}{\Delta t} - \nabla \cdot (\sigma \nabla u_{\boldsymbol{p}}^n) - \beta I_{total,\tilde{p}}^{n-1} \Big|_{\epsilon_i},$$

and let

$$\left[\frac{\partial u_{\boldsymbol{p}}^n}{\partial \hat{n}}\right]_\gamma := (\sigma \nabla u_{\boldsymbol{p}}^n)_\epsilon \cdot \hat{n}_\epsilon + (\sigma \nabla u_{\boldsymbol{p}}^n)_\xi \cdot \hat{n}_\xi$$

be the jump in the flux in the finite element solution across the edge (face in 3D) $\gamma = \epsilon \cap \xi$ shared by the two elements ϵ and ξ, with $\hat{n}_\epsilon$, $\hat{n}_\xi$ their respective outward-pointing unit normals along γ; note that when $\gamma \subset \partial\Omega$ we only have one of the two summands. Then the error $e_{\boldsymbol{p}}^n$ introduced by updating the fully discrete solution across one time-step from time t_{n-1} to time t_n as measured using E can be bounded as

$$\begin{aligned}
\sqrt{\int_{t_{n-1}}^{t_n} E(e_{\boldsymbol{p}}^n, e_{\boldsymbol{p}}^n) dt} \;\leq\; & C\Delta t \sum_{\epsilon_i \in \mathcal{M}} \left(\|r_n^i\|_{0,2,\epsilon_i}^2 \frac{h_{\epsilon_i}^2}{p_{\epsilon_i,n}^2} + \sum_{\gamma \in \partial\epsilon_i} \delta_\gamma \left\| \left[\frac{\partial u_{\boldsymbol{p}}^n}{\partial \hat{n}}\right]_\gamma \right\|_{0,2,\gamma}^2 \frac{h_\gamma}{p_{\gamma,n}} \right)^{\frac{1}{2}} \\
+ \; & C \left(\Delta t^{\frac{3}{2}} \sqrt{\frac{\beta}{C_m}} \left(\sum_{j=1}^m \left\| w_{\tilde{p},j}^n - w_j^n \right\|_{0,2,\Omega} + \left\| \Pi_{\tilde{p}}^{L2} I_{total}^{n-1} - I_{total}^{n-1} \right\|_{0,2,\Omega} \right) \right. \\
+ \; & \left. \left(\Delta t^{\frac{1}{2}} \sqrt{\beta C_m} + \Delta t^2 \sqrt{\beta / C_m} \right) \left\| e_{\boldsymbol{p}}^{n-1} \right\|_{0,2,\Omega} \right);
\end{aligned}$$

Proof. We follow the ideas of [137], proceeding to find a bound for $\sqrt{E(e_{\boldsymbol{p}}^n, e_{\boldsymbol{p}}^n)}$ by writing $\forall \chi \in \mathcal{H}^1(\Omega)$,

$$\begin{aligned}
E(e_{\boldsymbol{p}}^n, \chi) &= E(u_{\boldsymbol{p}}^n, \chi) - E(u^n, \chi) \\
&= E(u_{\boldsymbol{p}}^n, \chi) - \beta(\Delta t I_{total}^{n-1} + C_m u^{n-1}, \chi)_\Omega.
\end{aligned}$$

Splitting up the integrals $(\cdot,\cdot)_\Omega$ over the elements ϵ_i of the mesh $\mathcal{M}$, we can write the right-hand side as

$$\begin{aligned}
E(u_{\boldsymbol{p}}^n, \chi) - \beta(\Delta t I_{total}^{n-1} + C_m u^{n-1}, \chi)_\Omega \;=\; & \sum_{\epsilon_i \in \mathcal{M}} \Big[\beta C_m \left(u_{\boldsymbol{p}}^n, \chi\right)_{\epsilon_i} + \Delta t \left(\sigma \nabla u_{\boldsymbol{p}}^n, \nabla \chi\right)_{\epsilon_i} \\
- \; & \beta \left(\Delta t I_{total,\tilde{p}}^{n-1} + C_m u_{\boldsymbol{p}}^{n-1}, \chi\right)_{\epsilon_i} \Big] \\
+ \; & \beta \left(\Delta t \left(I_{total,\tilde{p}}^{n-1} - I_{total}^{n-1}\right) + C_m \left(u_{\boldsymbol{p}}^{n-1} - u^{n-1}\right), \chi\right)_\Omega.
\end{aligned}$$

Integrating by parts on each element ϵ_i, we obtain

$$\begin{aligned} E(e^n_{\boldsymbol{p}},\chi) &= \sum_{\epsilon_i\in\mathcal{M}}\Big[\beta C_m\left(u^n_{\boldsymbol{p}},\chi\right)_{\epsilon_i}+\Delta t\left\langle\sigma\nabla u^n_{\boldsymbol{p}}\cdot\hat{n}_{\epsilon_i},\chi\right\rangle_{\partial\epsilon_i}-\Delta t\left(\nabla\cdot(\sigma\nabla u^n_{\boldsymbol{p}}),\chi\right)_{\epsilon_i} \\ &- \beta\left(\Delta t I^{n-1}_{total,\tilde{p}}+C_m u^{n-1}_{\boldsymbol{p}},\chi\right)_{\epsilon_i}\Big] \\ &+ \beta\left(\Delta t\left(I^{n-1}_{total,\tilde{p}}-I^{n-1}_{total}\right)+C_m\left(u^{n-1}_{\boldsymbol{p}}-u^{n-1}\right),\chi\right)_{\Omega}, \end{aligned}$$

with $\partial\epsilon_i$ the boundary of the element ϵ_i and $\hat{n}_{\epsilon_i}$ the outward-pointing unit normal to $\partial\epsilon_i$, defined on faces of the element. Rewriting,

$$\begin{aligned} E(e^n_{\boldsymbol{p}},\chi) &= \Delta t\sum_{\epsilon_i\in\mathcal{M}}\left(\left(\beta C_m\frac{u^n_{\boldsymbol{p}}-u^{n-1}_{\boldsymbol{p}}}{\Delta t}-\nabla\cdot(\sigma\nabla u^n_{\boldsymbol{p}})-\beta I^{n-1}_{total,\tilde{p}},\chi\right)_{\epsilon_i}+\left\langle(\sigma\nabla u^n_{\boldsymbol{p}})_{\epsilon_i}\cdot\hat{n}_{\epsilon_i},\chi\right\rangle_{\partial\epsilon_i}\right) \\ &+ \beta\left(\Delta t\left(I^{n-1}_{total,\tilde{p}}-I^{n-1}_{total}\right)+C_m\left(u^{n-1}_{\boldsymbol{p}}-u^{n-1}\right),\chi\right)_{\Omega}. \end{aligned}$$

Rewriting the edge integral as a sum,

$$\begin{aligned} E(e^n_{\boldsymbol{p}},\chi) &= \Delta t\sum_{\epsilon_i\in\mathcal{M}}\left(r^i_n,\chi\right)_{\epsilon_i}+\Delta t\sum_{\gamma\in\partial\mathcal{M}}\left\langle\left[\frac{\partial u^n_{\boldsymbol{p}}}{\partial\hat{n}}\right]_{\gamma},\chi\right\rangle_{\gamma} \\ &+ \beta\left(\Delta t\left(I^{n-1}_{total,\tilde{p}}-I^{n-1}_{total}\right)+C_m\left(u^{n-1}_{\boldsymbol{p}}-u^{n-1}\right),\chi\right)_{\Omega}, \end{aligned}$$

where $\partial\mathcal{M}$ is the set of all element faces in $\mathcal{M}$ when the domain is three-dimensional, and the set of all element edges when it is two-dimensional.

Recall that $\Pi^n_{h\boldsymbol{p}}$ is a projection into $S^n_{h\boldsymbol{p}}$, and consider

$$\begin{aligned} E(e^n_{\boldsymbol{p}},\chi)-E(e^n_{\boldsymbol{p}},\Pi^n_{h\boldsymbol{p}}\chi) &= \Delta t\sum_{\epsilon_i\in\mathcal{M}}\left(r^i_n,\chi-\Pi^n_{h\boldsymbol{p}}\chi\right)_{\epsilon_i}+\Delta t\sum_{\gamma\in\partial\mathcal{M}}\left\langle\left[\frac{\partial u^n_{\boldsymbol{p}}}{\partial\hat{n}}\right]_{\gamma},\chi-\Pi^n_{h\boldsymbol{p}}\chi\right\rangle_{\gamma} \\ &+ \beta\left(\Delta t\left(I^{n-1}_{total,\tilde{p}}-I^{n-1}_{total}\right)+C_m\left(u^{n-1}_{\boldsymbol{p}}-u^{n-1}\right),\chi-\Pi^n_{h\boldsymbol{p}}\chi\right)_{\Omega}. \end{aligned}$$

Making use of the fact that $\forall\chi_{\boldsymbol{p}}\in S^n_{h\boldsymbol{p}}$,

$$E\left(e^n_{\boldsymbol{p}},\chi_{\boldsymbol{p}}\right)=\beta\Delta t\left(I^{n-1}_{total,\tilde{p}}-I^{n-1}_{total},\chi_{\boldsymbol{p}}\right)_{\Omega}+\beta C_m\left(u^{n-1}_{\boldsymbol{p}}-u^{n-1},\chi_{\boldsymbol{p}}\right)_{\Omega},$$

we see that

$$\begin{aligned}
E(e^n_{\boldsymbol{p}},\chi) &= \Delta t \sum_{\epsilon_i\in\mathcal{M}} \left(r^i_n, \chi - \Pi^n_{h\boldsymbol{p}}\chi\right)_{\epsilon_i} + \Delta t \sum_{\gamma\in\partial\mathcal{M}} \left\langle \left[\frac{\partial u^n_{\boldsymbol{p}}}{\partial\hat{n}}\right]_\gamma, \chi - \Pi^n_{h\boldsymbol{p}}\chi \right\rangle_\gamma \\
&+ \beta\left(\Delta t\left(I^{n-1}_{total,\tilde{p}} - I^{n-1}_{total}\right) + C_m\left(u^{n-1}_{\boldsymbol{p}} - u^{n-1}\right), \chi - \Pi^n_{h\boldsymbol{p}}\chi\right)_\Omega \\
&+ \beta\left(\Delta t\left(I^{n-1}_{total,\tilde{p}} - I^{n-1}_{total}\right) + C_m\left(u^{n-1}_{\boldsymbol{p}} - u^{n-1}\right), \Pi^n_{h\boldsymbol{p}}\chi\right)_\Omega, \\
E(e^n_{\boldsymbol{p}},\chi) &= \Delta t \sum_{\epsilon_i\in\mathcal{M}} \left(r^i_n, \chi - \Pi^n_{h\boldsymbol{p}}\chi\right)_{\epsilon_i} + \Delta t \sum_{\gamma\in\partial\mathcal{M}} \left\langle \left[\frac{\partial u^n_{\boldsymbol{p}}}{\partial\hat{n}}\right]_\gamma, \chi - \Pi^n_{h\boldsymbol{p}}\chi \right\rangle_\gamma \qquad (6.8) \\
&+ \beta\left(\Delta t\left(I^{n-1}_{total,\tilde{p}} - I^{n-1}_{total}\right) + C_m\left(u^{n-1}_{\boldsymbol{p}} - u^{n-1}\right), \chi\right)_\Omega.
\end{aligned}$$

Applying Cauchy-Schwarz and using definition (6.7),

$$\begin{aligned}
E(e^n_{\boldsymbol{p}},\chi) &\leq \Delta t \sum_{\epsilon_i\in\mathcal{M}} \|r^i_n\|_{0,2,\epsilon_i}\|\chi - \Pi^n_{h\boldsymbol{p}}\chi\|_{0,2,\epsilon_i} \\
&+ \Delta t \sum_{\gamma\in\partial\mathcal{M}} \left\|\left[\frac{\partial u^n_{\boldsymbol{p}}}{\partial\hat{n}}\right]_\gamma\right\|_{0,2,\gamma} \|\chi - \Pi^n_{h\boldsymbol{p}}\chi\|_{0,2,\gamma} \\
&+ \left(\Delta t A_{n-1} + \beta C_m \left\|e^{n-1}_{\boldsymbol{p}}\right\|_{0,2,\Omega}\right)\|\chi\|_{0,2,\Omega},
\end{aligned}$$

from which, together with inequalities (6.6), we obtain

$$\begin{aligned}
E(e^n_{\boldsymbol{p}},\chi) &\leq C\Delta t \sum_{\epsilon_i\in\mathcal{M}} \|r^i_n\|_{0,2,\epsilon_i} \frac{h_{\epsilon_i}}{p_{\epsilon_i,n}} \|\nabla\chi\|_{0,2,\omega^1_{\epsilon_i}} \qquad (6.9) \\
&+ \Delta t \sum_{\gamma\in\partial\mathcal{M}} \left\|\left[\frac{\partial u^n_{\boldsymbol{p}}}{\partial\hat{n}}\right]_\gamma\right\|_{0,2,\gamma} \sqrt{\frac{h_\gamma}{p_{\gamma,n}}} \|\nabla\chi\|_{0,2,\omega^1_\gamma} \\
&+ \left(\Delta t A_{n-1} + \beta C_m \left\|e^{n-1}_{\boldsymbol{p}}\right\|_{0,2,\Omega}\right)\|\chi\|_{0,2,\Omega};
\end{aligned}$$

For $\lambda_{\min}(\sigma)$ the smallest eigenvalue of σ, we have

$$\begin{aligned}
\|\nabla\chi\|^2_{0,2,\omega^1_{\epsilon_i}} = (\nabla\chi,\nabla\chi)_{\omega^1_{\epsilon_i}} &\leq \frac{1}{\lambda_{\min}(\sigma)}\left(\sigma^{\frac{1}{2}}\nabla\chi, \sigma^{\frac{1}{2}}\nabla\chi\right)_{\omega^1_{\epsilon_i}} \\
&= \frac{1}{\Delta t\lambda_{\min}(\sigma)}\left[\Delta t\left(\sigma\nabla\chi,\nabla\chi\right)_{\omega^1_{\epsilon_i}}\right] \\
&\leq E_{\omega^1_{\epsilon_i}}(\chi,\chi)/(\Delta t\lambda_{\min}(\sigma)),
\end{aligned}$$

where we have assigned a subscript to $E_{\omega^1_{\epsilon_i}}$ in order to make it clear that we are considering it on a patch $\omega^1_{\epsilon_i}$ here. Because of this, and with $s : \mathcal{M} \cup \partial\mathcal{M} \to \mathcal{M}$ some fixed function which is the identity on $\mathcal{M}$ and for $\gamma \in \partial\mathcal{M}$ takes the value $s(\gamma) = \epsilon \in \mathcal{M}$ for some arbitrary ϵ such that $\gamma \subset \partial\epsilon$, we see that the bound

$$\begin{aligned}
\sqrt{\sum_{y\in\mathcal{M}\cup\partial\mathcal{M}} \|\nabla\chi\|^2_{0,2,\omega^1_{s(y)}}} &\leq \sqrt{\sum_{y\in\mathcal{M}\cup\partial\mathcal{M}} \frac{E_{\omega^1_{s(y)}}(\chi,\chi)}{\Delta t\lambda_{\min}(\sigma)}} \\
&\leq \sum_{y\in\mathcal{M}\cup\partial\mathcal{M}} \frac{\sqrt{E_{\omega^1_{s(y)}}(\chi,\chi)}}{\sqrt{\Delta t\lambda_{\min}(\sigma)}} \\
&\leq \frac{k}{\sqrt{\Delta t\lambda_{\min}(\sigma)}}\sqrt{E(\chi,\chi)} \qquad (6.10)
\end{aligned}$$

holds for some k equal to the maximum number of times any one element is covered by the patches $w^1_{s(y)}$ over $\{y : y \in \mathcal{M} \cup \partial\mathcal{M}\}$. Applying the Cauchy-Schwarz inequality to Inequality (6.9), noticing that $\omega^1_\gamma \subset \omega^1_{s(\gamma)}$ so that $\|\nabla\chi\|_{0,2,\omega^1_\gamma} \leq \|\nabla\chi\|_{0,2,\omega^1_{s(\gamma)}}$ and then applying Inequality (6.10), we get

$$\begin{aligned}
E(e^n_{\boldsymbol{p}},\chi) &\leq C\Delta t^{\frac{1}{2}}\sqrt{E(\chi,\chi)}\left(\sum_{\epsilon_i\in\mathcal{M}} \|r^i_n\|^2_{0,2,\epsilon_i}\frac{h^2_{\epsilon_i}}{p^2_{\epsilon_i,n}} + \sum_{\gamma\in\partial\mathcal{M}}\left\|\left[\frac{\partial u^n_{\boldsymbol{p}}}{\partial\hat{n}}\right]_\gamma\right\|^2_{0,2,\gamma}\frac{h_\gamma}{p_{\gamma,n}}\right)^{\frac{1}{2}} \\
&+ \left(\Delta t A_{n-1} + \beta C_m\left\|e^{n-1}_{\boldsymbol{p}}\right\|_{0,2,\Omega}\right)\|\chi\|_{0,2,\Omega},
\end{aligned}$$

where we have absorbed $k/\sqrt{\lambda_{\min}(\sigma)}$ into C. Taking $\chi = e^n_{\boldsymbol{p}}$, dividing through by $\sqrt{\Delta t E(e^n_{\boldsymbol{p}}, e^n_{\boldsymbol{p}})}$ and integrating this in time over the time interval $[t_{n-1}, t_n) := [(n-1)\Delta t, n\Delta t)$, we can write

$$\begin{aligned}
\sqrt{\int_{t_{n-1}}^{t_n} E(e^n_{\boldsymbol{p}}, e^n_{\boldsymbol{p}})dt} &\leq C\Delta t\left(\sum_{\epsilon_i\in\mathcal{M}} \|r^i_n\|^2_{0,2,\epsilon_i}\frac{h^2_{\epsilon_i}}{p^2_{\epsilon_i,n}} + \sum_{\gamma\in\partial\mathcal{M}}\left\|\left[\frac{\partial u^n_{\boldsymbol{p}}}{\partial\hat{n}}\right]_\gamma\right\|^2_{0,2,\gamma}\frac{h_\gamma}{p_{\gamma,n}}\right)^{\frac{1}{2}} \qquad (6.11) \\
&+ C\left(\Delta t^{\frac{3}{2}}A_{n-1} + \Delta t^{\frac{1}{2}}\beta C_m\left\|e^{n-1}_{\boldsymbol{p}}\right\|_{0,2,\Omega}\right)\|e^n_{\boldsymbol{p}}\|_{0,2,\Omega}\Big/\sqrt{E(e^n_{\boldsymbol{p}}, e^n_{\boldsymbol{p}})},
\end{aligned}$$

because our expressions are piecewise-constant in time on these intervals. We can use the

fact that $\sqrt{E(e^n_{\boldsymbol{p}}, e^n_{\boldsymbol{p}})} \geq \sqrt{\beta C_m}\|e^n_{\boldsymbol{p}}\|_{0,2,\Omega}$ and re-arrange the term in the inner brackets to obtain

$$\begin{aligned}\sqrt{\int_{t_{n-1}}^{t_n} E(e^n_{\boldsymbol{p}}, e^n_{\boldsymbol{p}})dt} &\leq C\Delta t \left(\sum_{\epsilon_i \in \mathcal{M}} \left(\|r^i_n\|^2_{0,2,\epsilon_i} \frac{h^2_{\epsilon_i}}{p^2_{\epsilon_i,n}} + \sum_{\gamma \in \partial\epsilon_i} \delta_\gamma \left\| \left[\frac{\partial u^n_{\boldsymbol{p}}}{\partial \hat{n}} \right]_\gamma \right\|^2_{0,2,\gamma} \frac{h_\gamma}{p_{\gamma,n}} \right) \right)^{\frac{1}{2}} \\ &+ C\left(\Delta t^{\frac{3}{2}} (\beta C_m)^{-\frac{1}{2}} A_{n-1} + \Delta t^{\frac{1}{2}} (\beta C_m)^{\frac{1}{2}} \left\| e^{n-1}_{\boldsymbol{p}} \right\|_{0,2,\Omega} \right), \end{aligned} \tag{6.12}$$

where the inner sum is over the edges (in 2D; faces in 3D) of the element ϵ_i and

$$\delta_\gamma = \begin{cases} 1, & \gamma \subset \partial\Omega \\ \frac{1}{2}, & \text{otherwise;} \end{cases}$$

this has been inserted into Equation (6.12) to compensate for the fact that the terms involving the edge normals now each occur twice over the outer sum whenever the edge (or face) γ is shared by two elements.

We now expand the term A_{n-1} using the L^2 projection $\Pi^{L2}_{\tilde{p}} : L^2(\Omega) \to S_{h\tilde{p}}$:

$$\begin{aligned} A_{n-1} \,/\, \beta &= \left\| I^{n-1}_{total,\tilde{p}} - I^{n-1}_{total} \right\|_{0,2,\Omega} \\ (def) &= \left\| \Pi^{L2}_{\tilde{p}} I_{total}\left(u^{n-1}_{\boldsymbol{p}}, w^n_{\tilde{p}}\right) - I_{total}(u^{n-1}, w^n) \right\|_{0,2,\Omega} \\ &\leq \left\| \Pi^{L2}_{\tilde{p}} I_{total}\left(u^{n-1}_{\boldsymbol{p}}, w^n_{\tilde{p}}\right) - \Pi^{L2}_{\tilde{p}} I_{total}(u^{n-1}, w^n) \right\|_{0,2,\Omega} \\ &+ \left\| \Pi^{L2}_{\tilde{p}} I_{total}(u^{n-1}, w^n) - I_{total}(u^{n-1}, w^n) \right\|_{0,2,\Omega}. \end{aligned}$$

By applying the property $\left\|\Pi^{L2}_{\tilde{p}}(v)\right\|_{0,2,\Omega} \leq \|v\|_{0,2,\Omega}$ of $\Pi^{L2}_{\tilde{p}}$ and using the Lipschitz property

$$\left\| I_{total}(u^{n-1}_{\boldsymbol{p}}, w^n_{\tilde{p}}) - I_{total}(u^{n-1}, w^n) \right\|_{0,2,\Omega} \leq L \left(\left\| u^{n-1}_{\boldsymbol{p}} - u^{n-1} \right\|_{0,2,\Omega} + \sum_{j=1}^{m} \left\| w^n_{\tilde{p},j} - w^n_j \right\|_{0,2,\Omega} \right) \tag{6.13}$$

of I_{total} with Lipschitz constant L (we will discuss this in Section 7.1.1), we can write

$$A_{n-1} \,/\, \beta \leq L \left(\left\| u_{\boldsymbol{p}}^{n-1} - u^{n-1} \right\|_{0,2,\Omega} + \sum_{j=1}^{m} \left\| w_{\tilde{p},j}^{n} - w_{j}^{n} \right\|_{0,2,\Omega} \right) + \left\| \Pi_{\tilde{p}}^{L2} I_{total}^{n-1} - I_{total}^{n-1} \right\|_{0,2,\Omega}, \tag{6.14}$$

which can then be inserted into Inequality (6.12), giving

$$\begin{aligned}
\sqrt{\int_{t_{n-1}}^{t_n} E(e_{\boldsymbol{p}}^n, e_{\boldsymbol{p}}^n) dt} \;\; \leq \;\; & C\Delta t \left(\sum_{\epsilon_i \in \mathcal{M}} \left(\| r_n^i \|_{0,2,\epsilon_i}^2 \frac{h_{\epsilon_i}^2}{p_{\epsilon_i,n}^2} + \sum_{\gamma \in \partial\epsilon_i} \delta_\gamma \left\| \left[\frac{\partial u_{\boldsymbol{p}}^n}{\partial \hat{n}} \right]_\gamma \right\|_{0,2,\gamma}^2 \frac{h_\gamma}{p_{\gamma,n}} \right) \right)^{\frac{1}{2}} \\
+ \;\; & C \left(\Delta t^{\frac{3}{2}} \sqrt{\frac{\beta}{C_m}} \left(\sum_{j=1}^{m} \left\| w_{\tilde{p},j}^n - w_j^n \right\|_{0,2,\Omega} + \left\| \Pi_{\tilde{p}}^{L2} I_{total}^{n-1} - I_{total}^{n-1} \right\|_{0,2,\Omega} \right) \right. \\
+ \;\; & \left. \left(\Delta t^{\frac{1}{2}} \sqrt{\beta C_m} + \Delta t^{\frac{3}{2}} \sqrt{\beta / C_m} \right) \left\| e_{\boldsymbol{p}}^{n-1} \right\|_{0,2,\Omega} \right);
\end{aligned}$$

Taking the square root inside the sum completes the proof. □

Alternatively, we can find a global-in-time error bound over the interval of simulation $[0, T]$ by employing a discrete Gronwall inequality, as we demonstrate in the following theorem.

Theorem 6.2.2. *Suppose $\Delta t < 1$. We can then bound the error introduced by the finite element scheme by time t_N measured in the $L^2(\Omega)$ norm as*

$$\begin{aligned}
\| e_{\boldsymbol{p}}^N \|_{0,2,\Omega} \leq & \left[\sqrt{\frac{\beta C_m^2 + \Delta t L}{\beta C_m^2 (1 - \Delta t)}} \| e_{\boldsymbol{p}}^0 \|_{0,2,\Omega} \right. \\
+ \;\; & \left. \sum_{n=1}^{N} \left(\sqrt{\frac{\Delta t D_n}{\beta C_m (1 - \Delta t)}} + \sqrt{\frac{C\Delta t}{\lambda_{min}(\sigma) \beta C_m (1 - \Delta t)}} \sum_{\epsilon_i \in \mathcal{M}} \eta_{\epsilon_i, n} \right) \right] \exp \left(\sum_{n=1}^{N-1} \frac{\Delta t}{2} \left(\frac{\beta C_m^2 + L}{\beta C_m^2 (1 - \Delta t)} \right) \right).
\end{aligned}$$

where

$$D_n := \frac{1}{C_m} \left(L \sum_{j=1}^{m} \left\| w_j^n - w_{\tilde{p},j}^n \right\|_{0,2,\Omega}^2 + \left\| \Pi_{\tilde{p}}^{L2} I_{total}^{n-1} - I_{total}^{n-1} \right\|_{0,2,\Omega}^2 \right).$$

Proof. Return to equation (6.8),

$$\begin{aligned} E(e_{\boldsymbol{p}}^n, \chi) &= \Delta t \left(\sum_{\epsilon_i \in \mathcal{M}} \left(r_n^i, \chi - \Pi_{h\boldsymbol{p}}^n \chi \right)_{\epsilon_i} + \sum_{\gamma \in \partial \mathcal{M}} \left\langle \left[\frac{\partial u_{\boldsymbol{p}}^n}{\partial \hat{n}} \right]_\gamma, \chi - \Pi_{h\boldsymbol{p}}^n \chi \right\rangle_\gamma \right) \\ &+ \beta \left(\Delta t \left(I_{total,\tilde{p}}^{n-1} - I_{total}^{n-1} \right) + C_m \left(u_{\boldsymbol{p}}^{n-1} - u^{n-1} \right), \chi \right)_\Omega, \end{aligned}$$

expand E and consider an alternate way of dealing with the term $e_{\boldsymbol{p}}^{n-1} = u_{\boldsymbol{p}}^{n-1} - u^{n-1}$ as follows:

$$\begin{aligned} \beta C_m (e_{\boldsymbol{p}}^n, \chi)_\Omega + \Delta t (\sigma \nabla e_{\boldsymbol{p}}^n, \nabla \chi)_\Omega &= \Delta t \left(\sum_{\epsilon_i \in \mathcal{M}} \left(r_n^i, \chi - \Pi_{h\boldsymbol{p}}^n \chi \right)_{\epsilon_i} + \sum_{\gamma \in \partial \mathcal{M}} \left\langle \left[\frac{\partial u_{\boldsymbol{p}}^n}{\partial \hat{n}} \right]_\gamma, \chi - \Pi_{h\boldsymbol{p}}^n \chi \right\rangle_\gamma \right) \\ &+ \beta \left(\Delta t \left(I_{total,\tilde{p}}^{n-1} - I_{total}^{n-1}, \chi \right)_\Omega + C_m (e_{\boldsymbol{p}}^{n-1}, \chi)_\Omega \right) \quad \forall \chi \in \mathcal{H}^1(\Omega). \end{aligned}$$

Applying Cauchy-Schwarz and Young's inequality,

$$\begin{aligned} \beta C_m (e_{\boldsymbol{p}}^n, \chi)_\Omega + \Delta t (\sigma \nabla e_{\boldsymbol{p}}^n, \nabla \chi)_\Omega &\leq \Delta t \left(\sum_{\epsilon_i \in \mathcal{M}} \left(r_n^i, \chi - \Pi_{h\boldsymbol{p}}^n \chi \right)_{\epsilon_i} + \sum_{\gamma \in \partial \mathcal{M}} \left\langle \left[\frac{\partial u_{\boldsymbol{p}}^n}{\partial \hat{n}} \right]_\gamma, \chi - \Pi_{h\boldsymbol{p}}^n \chi \right\rangle_\gamma \right) \\ &+ \beta \left(\Delta t \left(I_{total,\tilde{p}}^{n-1} - I_{total}^{n-1}, \chi \right)_\Omega + \frac{C_m \| e_{\boldsymbol{p}}^{n-1} \|_{0,2,\Omega}^2}{2} + \frac{C_m \| \chi \|_{0,2,\Omega}^2}{2} \right). \end{aligned}$$

Using inequalities (6.6), applying Cauchy-Schwarz and rearranging the sums as for Inequality (6.12) of the previous theorem, we obtain

$$\begin{aligned} \beta C_m (e_{\boldsymbol{p}}^n, \chi)_\Omega &+ \Delta t (\sigma \nabla e_{\boldsymbol{p}}^n, \nabla \chi)_\Omega \leq \\ & C \Delta t \| \nabla \chi \|_{0,2,\Omega} \left(\sum_{\epsilon_i \in \mathcal{M}} \left(\| r_n^i \|_{0,2,\epsilon_i}^2 \frac{h_{\epsilon_i}^2}{p_{\epsilon_i,n}^2} + \sum_{\gamma \in \partial \epsilon_i} \delta_\gamma \left\| \left[\frac{\partial u_{\boldsymbol{p}}^n}{\partial \hat{n}} \right]_\gamma \right\|_{0,2,\gamma}^2 \frac{h_\gamma}{p_{\gamma,n}} \right) \right)^{\frac{1}{2}} \\ &+ \Delta t \beta \| I_{total,\tilde{p}}^{n-1} - I_{total}^{n-1} \|_{0,2,\Omega} \| \chi \|_{0,2,\Omega} + \beta \left(\frac{C_m \| e_{\boldsymbol{p}}^{n-1} \|_{0,2,\Omega}^2}{2} + \frac{C_m \| \chi \|_{0,2,\Omega}^2}{2} \right). \end{aligned}$$

With $\chi = e_{\boldsymbol{p}}^n$ and two more applications of Young's inequality, this time with splitting

constants μ and ν, and bounding $\|I^{n-1}_{total,\tilde{p}} - I^{n-1}_{total}\|_{0,2,\Omega}$ as in Inequality (6.14), we have

$$\begin{aligned}
\beta C_m \|e^n_{\boldsymbol{p}}\|^2_{0,2,\Omega} \quad &+ \quad \Delta t \lambda_{min}(\sigma) \|\nabla e^n_{\boldsymbol{p}}\|^2_{0,2,\Omega} \leq \frac{C\Delta t}{2\mu} \sum_{\epsilon_i \in \mathcal{M}} \eta^2_{\epsilon_i,n} + \frac{C\Delta t \mu}{2} \|\nabla e^n_{\boldsymbol{p}}\|^2_{0,2,\Omega} \\
&+ \quad \Delta t \beta \frac{L \left\|e^{n-1}_{\boldsymbol{p}}\right\|^2 + L \sum_{j=1}^m \left\|w^n_j - w^n_{\tilde{p},j}\right\|^2 + \left\|\Pi^{L2}_{\tilde{p}} I^{n-1}_{total} - I^{n-1}_{total}\right\|^2}{2\nu} + \\
&+ \quad \Delta t \frac{\nu \|e^n_{\boldsymbol{p}}\|^2_{0,2,\Omega}}{2} + \frac{\beta C_m \|e^{n-1}_{\boldsymbol{p}}\|^2_{0,2,\Omega}}{2} + \frac{\beta C_m \|e^n_{\boldsymbol{p}}\|^2_{0,2,\Omega}}{2}.
\end{aligned}$$

Taking $\mu = 2\lambda_{min}(\sigma)/C$ and rearranging,

$$\begin{aligned}
\beta C_m \left(\|e^n_{\boldsymbol{p}}\|^2_{0,2,\Omega} - \|e^{n-1}_{\boldsymbol{p}}\|^2_{0,2,\Omega}\right) \quad &\leq \quad \frac{C\Delta t}{\lambda_{min}(\sigma)} \sum_{\epsilon_i \in \mathcal{M}} \eta^2_{\epsilon_i,n} \\
&+ \quad \Delta t \nu \|e^n_{\boldsymbol{p}}\|^2_{0,2,\Omega} + \frac{\Delta t \beta L}{\nu} \|e^{n-1}_{\boldsymbol{p}}\|^2_{0,2,\Omega} \\
&+ \quad \frac{\Delta t \beta}{\nu} \left(L \sum_{j=1}^m \left\|w^n_j - w^n_{\tilde{p},j}\right\|^2 + \left\|\Pi^{L2}_{\tilde{p}} I^{n-1}_{total} - I^{n-1}_{total}\right\|^2 \right),
\end{aligned}$$

then summing over n ranging from 1 to N,

$$\begin{aligned}
\beta C_m \left(\|e^N_{\boldsymbol{p}}\|^2_{0,2,\Omega} - \|e^0_{\boldsymbol{p}}\|^2_{0,2,\Omega}\right) \quad &\leq \quad \frac{C\Delta t}{\lambda_{min}(\sigma)} \sum_{n=1}^N \sum_{\epsilon_i \in \mathcal{M}} \eta^2_{\epsilon_i,n} + \Delta t \nu \sum_{n=1}^N \|e^n_{\boldsymbol{p}}\|^2_{0,2,\Omega} \\
&+ \quad \frac{\Delta t \beta L}{\nu} \sum_{n=1}^N \|e^{n-1}_{\boldsymbol{p}}\|^2_{0,2,\Omega} \\
&+ \quad \frac{\Delta t \beta}{\nu} \sum_{n=1}^N \left(L \sum_{j=1}^m \left\|w^n_j - w^n_{\tilde{p},j}\right\|^2 + \left\|\Pi^{L2}_{\tilde{p}} I^{n-1}_{total} - I^{n-1}_{total}\right\|^2 \right).
\end{aligned}$$

This can be rearranged and re-indexed to give

$$\begin{aligned}
\left(\beta C_m - \Delta t \nu\right) \|e^N_{\boldsymbol{p}}\|^2_{0,2,\Omega} \quad &\leq \quad \frac{C\Delta t}{\lambda_{min}(\sigma)} \left(\sum_{n=1}^{N-1} \sum_{\epsilon_i \in \mathcal{M}} \eta^2_{\epsilon_i,n+1} + \sum_{\epsilon_i \in \mathcal{M}} \eta^2_{\epsilon_i,1} \right) \\
&+ \quad \left(\Delta t \nu + \frac{\Delta t \beta L}{\nu} \right) \sum_{n=1}^{N-1} \|e^n_{\boldsymbol{p}}\|^2_{0,2,\Omega} \\
&+ \quad \frac{\beta C_m \Delta t}{\nu} \left(\sum_{n=1}^{N-1} D_{n+1} + D_1 \right) + \left(\beta C_m + \frac{\Delta t \beta L}{\nu} \right) \|e^0_{\boldsymbol{p}}\|^2_{0,2,\Omega}.
\end{aligned}$$

Choosing $\nu = \beta C_m$,

$$\begin{aligned}\beta C_m\left(1-\Delta t\right)\|e_{\boldsymbol{p}}^N\|_{0,2,\Omega}^2 &\leq \Delta t D_1 + \left(\beta C_m + \frac{\Delta t L}{C_m}\right)\|e_{\boldsymbol{p}}^0\|_{0,2,\Omega}^2 + \frac{C\Delta t}{\lambda_{min}(\sigma)}\sum_{\epsilon_i\in\mathcal{M}}\eta_{\epsilon_i,1}^2 \\ &+ \sum_{n=1}^{N-1}\left(\Delta t D_{n+1} + \frac{C\Delta t}{\lambda_{min}(\sigma)}\sum_{\epsilon_i\in\mathcal{M}}\eta_{\epsilon_i,n+1}^2\right) \\ &+ \sum_{n=1}^{N-1}\Delta t\left(\beta C_m + \frac{L}{C_m}\right)\|e_{\boldsymbol{p}}^n\|_{0,2,\Omega}^2,\end{aligned}$$

which upon dividing by $\beta C_m\left(1-\Delta t\right)$ satisfies the conditions for the discrete Gronwall inequality [113] p.14, giving

$$\begin{aligned}\|e_{\boldsymbol{p}}^N\|_{0,2,\Omega}^2 &\leq \exp\left(\sum_{n=1}^{N-1}\Delta t\left(\frac{\beta C_m^2 + L}{\beta C_m^2(1-\Delta t)}\right)\right)\left[\frac{\Delta t D_1}{\beta C_m(1-\Delta t)} + \left(\frac{\beta C_m^2 + \Delta t L}{\beta C_m^2(1-\Delta t)}\right)\|e_{\boldsymbol{p}}^0\|_{0,2,\Omega}^2\right. \\ &+ \frac{C\Delta t}{\lambda_{min}(\sigma)\beta C_m(1-\Delta t)}\sum_{\epsilon_i\in\mathcal{M}}\eta_{\epsilon_i,1}^2 \\ &+ \left.\sum_{n=1}^{N-1}\left(\frac{\Delta t D_{n+1}}{\beta C_m(1-\Delta t)} + \frac{C\Delta t}{\lambda_{min}(\sigma)\beta C_m(1-\Delta t)}\sum_{\epsilon_i\in\mathcal{M}}\eta_{\epsilon_i,n+1}^2\right)\right],\end{aligned}$$

or re-indexing again,

$$\begin{aligned}\|e_{\boldsymbol{p}}^N\|_{0,2,\Omega}^2 &\leq \exp\left(\sum_{n=1}^{N-1}\Delta t\left(\frac{\beta C_m^2 + L}{\beta C_m^2(1-\Delta t)}\right)\right)\left[\left(\frac{\beta C_m^2 + \Delta t L}{\beta C_m^2(1-\Delta t)}\right)\|e_{\boldsymbol{p}}^0\|_{0,2,\Omega}^2\right. \\ &+ \left.\sum_{n=1}^{N}\left(\frac{\Delta t D_n}{\beta C_m(1-\Delta t)} + \frac{C\Delta t}{\lambda_{min}(\sigma)\beta C_m(1-\Delta t)}\sum_{\epsilon_i\in\mathcal{M}}\eta_{\epsilon_i,n}^2\right)\right].\end{aligned}$$

this can be square-rooted to give the required conclusion. □

6.3 Adaptive Strategy

Here we give the algorithm that we use to choose the polynomial degree distribution $\boldsymbol{p}$ on the n-th time-step, given the computed error indicators $\eta_{\epsilon_i,n}$. First, we need an expression for the magnitude of the error that we currently have. Theoretical and uniform-degree empirical results tell us that we can expect the error in an element ϵ_i to be an exponential

function of its polynomial degree p_i when the true solution is smooth [78, 21], which we assume for our problem. Thus we express it as

$$E_{\epsilon_i} \propto \exp(-\omega p_i) \tag{6.15}$$

for a constant ω which we can estimate from observed convergence rates for non-adaptive formulations of high-order elements for the monodomain problem, for example the data displayed in Figure 5.6(b); we use ω=1.66. We also need an expression for the size of the error that we wish to have; suppose that this target is E_{T,ϵ_i}. Again, we assume this to be exponential in the target polynomial degree q_i that we are to determine, and so we write

$$E_{T,\epsilon_i} \propto \exp(-\omega q_i). \tag{6.16}$$

Combining Equations (6.15) and (6.16), we see that we should take

$$q_i = p_i + \frac{\ln(E_{\epsilon_i}/E_{T,\epsilon_i})}{\omega} \tag{6.17}$$

for the new polynomial degree in element ϵ_i.

Since we do not know E_{ϵ_i}, here we use the approximation $E_{\epsilon_i} \simeq \eta_{\epsilon_i,n}$, which we compute for each element. Inequality (6.4) shows this approximation makes sense, and we have support from the fact that it works very effectively in practice; see for example Figure 6.1.

Because the value of q_i that Equation (6.17) produces may not be an integer or may not lie in the range of element degrees $1, 2, \ldots, p$ our scheme is capable of using, we round to the nearest integer $\hat{q}_i \geq q_i$ and clamp to the interval $[1, p]$.

We employ the error indicator on each time-step of our solution scheme as follows:

1. Compute the Lagrange finite element solution for the non-diffusing cell state variables w. Compute the current field $I^{n-1}_{total,\tilde{p}}$ from this.

2. Perform a pre-solve for the transmembrane potential, using p=1 finite elements everywhere in the domain and generating $u_{\mathbf{1}}^n$; this means that we must take $p_i = 1 \ \forall \ i$ in Equation (6.17). The output of this step is not used as an update to the transmembrane potential; it purpose is to provide an up-to-date snapshot of the location of significant spatial error. This step is cheap due to the fixed coarse mesh that the high-order elements allow us to use.

3. Taking $u_{\boldsymbol{p}}^n$ in Equation (6.3) to be $u_{\mathbf{1}}^n$, compute the error estimate for each element. Here, we choose some small positive percentage error tolerance θ and take our target error to be

$$E_{T,\epsilon_i} = \frac{\theta}{100}\sqrt{\int_{t_{n-1}}^{t_n} E(u_{\mathbf{1}}^n, u_{\mathbf{1}}^n)dt},$$

where we divide by 100 to convert θ from a percentage to a fraction, so that the error is relative to the magnitude of the solution. Here, $t_n = n\Delta t$.

4. Adapt the basis degrees for all the elements in the mesh accordingly, setting the degree of ϵ_i to be the newly found $\hat{q}_i$. In order to perform the actual adaptation, we work from stored reference mass and stiffness matrices constructed with every element at its maximum possible degree. From these, we extract the rows and columns corresponding to the basis functions which the system has chosen to use at the current point in time; this gives us our adapted matrices without having to perform any costly calculation.

5. Solve our adapted system for $\boldsymbol{u}_{\boldsymbol{p}}^n$ (with $\boldsymbol{p} = (\hat{q}_1, \hat{q}_2, \ldots, \hat{q}_N)$), using conjugate gradients and taking the result of Step 2 as an initial guess.

It is possible that two neighbouring elements ϵ_i and ϵ_j will have different degrees assigned to them. In this situation it is not immediately obvious what should be done with a basis function of degree s whose support includes both ϵ_i and ϵ_j if, for example, $\hat{q}_i < s \leq \hat{q}_j$. In such circumstances, we make the rule that we include this basis function, even though it is not required by one of the elements that it belongs to. Thus we err

on the side of caution, choosing to make our approximate solution slightly over-accurate rather than under-accurate.

6.4 The Adapted Linear System

Let $\mathcal{P}_n$ be the set of basis functions which we use on the n-th time-step to represent $u_{\boldsymbol{p}}^n$ and let $\mathcal{P}_{max}$ be the set of all basis functions when all elements are at their maximal degree. Define $N_n := |\mathcal{P}_n|$ $N_{max} := |\mathcal{P}_{max}|$, and let $\tilde{\boldsymbol{u}}_{\boldsymbol{p}}^{n-1}$ be the extension of $\boldsymbol{u}_{\boldsymbol{p}}^{n-1}$ to length N_{max} by including zero coefficients for the basis functions in $\mathcal{P}_{max} \setminus \mathcal{P}_{n-1}$. Under the adaptive discretisation, the the fully discrete form of the monodomain equation (5.1a) is

$$\left(\boldsymbol{M} + \frac{\Delta t}{\beta C_m}\boldsymbol{A}\right)\boldsymbol{u}_{\boldsymbol{p}}^n = \boldsymbol{M}_R\left(\tilde{\boldsymbol{u}}_{\boldsymbol{p}}^{n-1} + \frac{\Delta t}{C_m}\boldsymbol{I}_{total}^{n-1}\right);$$

where $\boldsymbol{M}$ and $\boldsymbol{A}$ are now $N_n \times N_n$ adapted forms of the maximal-degree mass and stiffness reference matrices; the former are constructed on each time-step by extracting from the latter the rows and columns corresponding to the in-use basis functions, as identified by the error estimation procedure. We have the second mass matrix $\boldsymbol{M}_R$ on the right-hand side; this must now be a rectangular matrix of size $N_n \times N_{max}$, which we construct by extracting complete rows from the reference mass matrix for the in-use basis functions. Doing this means that we can integrate the never-adapted, uniform-degree $\tilde{p}$ current term $\boldsymbol{I}_{total}^{n-1}$; note that it also allows us to integrate the full state of the system on the previous time-step $\tilde{\boldsymbol{u}}_{\boldsymbol{p}}^{n-1}$ despite the fact that the basis may have changed since then.

6.4.1 Preconditioning in 2D

Once we have performed the row and column extraction and have our adapted system matrix $\boldsymbol{Q} = \boldsymbol{M} + \frac{\Delta t}{\beta C_m}\boldsymbol{A}$ for a particular time-step, we can write it as

$$\boldsymbol{Q} = \begin{pmatrix} \boldsymbol{Q}_{LL} & \boldsymbol{Q}_{LH}^T \\ \boldsymbol{Q}_{LH} & \boldsymbol{Q}_{HH} \end{pmatrix},$$

where $\boldsymbol{Q}_{LL}$, $\boldsymbol{Q}_{LH}$ and $\boldsymbol{Q}_{HH}$ are the linear-linear, linear-(higher degree) and (higher degree)-(higher degree) basis function interaction blocks, respectively. Due to Step 4 of the adaptive algorithm described in Section 6.3, we see that the blocks $\boldsymbol{Q}_{HH}$ and $\boldsymbol{Q}_{LH}$ will generally be small due to the large number of basis functions that will be "switched off" by the adaptivity; this is the main source of CPU time savings. Note that the block $\boldsymbol{Q}_{LL}$ never changes in size because the linear basis functions are always present in $S_{h\boldsymbol{p}}$.

Recall that we use PCG to solve our linear system (5.14) on each iteration. As a preconditioner, in two dimensional studies we use a 2×2 block LDU decomposition of our system matrix, choosing the linear basis function entries as the upper-left block and the higher-order basis functions as the lower-right block, similar to the block decomposition approach that has been used to split transmembrane and extracellular potentials in bidomain simulations [13]. For completeness, we reproduce the decomposition here.

The matrix $\boldsymbol{Q}$ is factorized as

$$\boldsymbol{Q} = \begin{pmatrix} \boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{Q}_{LH}\boldsymbol{Q}_{LL}^{-1} & \boldsymbol{I} \end{pmatrix} \begin{pmatrix} \boldsymbol{Q}_{LL} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{S} \end{pmatrix} \begin{pmatrix} \boldsymbol{I} & \boldsymbol{Q}_{LL}^{-1}\boldsymbol{Q}_{LH}^{T} \\ \boldsymbol{0} & \boldsymbol{I} \end{pmatrix},$$

with $\boldsymbol{S} := \boldsymbol{Q}_{HH} - \boldsymbol{Q}_{LH}\boldsymbol{Q}_{LL}^{-1}\boldsymbol{Q}_{LH}^{T}$ the Schur complement. We can then write

$$\boldsymbol{Q}^{-1} = \begin{pmatrix} \boldsymbol{I} & -\boldsymbol{Q}_{LL}^{-1}\boldsymbol{Q}_{LH}^{T} \\ \boldsymbol{0} & \boldsymbol{I} \end{pmatrix} \begin{pmatrix} \boldsymbol{Q}_{LL}^{-1} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{S}^{-1} \end{pmatrix} \begin{pmatrix} \boldsymbol{I} & \boldsymbol{0} \\ -\boldsymbol{Q}_{LH}\boldsymbol{Q}_{LL}^{-1} & \boldsymbol{I} \end{pmatrix}.$$

We make the approximation $\boldsymbol{S}^{-1} \approx \boldsymbol{Q}_{HH}^{-1}$ and then use the resulting approximate decomposition of $\boldsymbol{Q}^{-1}$ as

$$\begin{pmatrix} \boldsymbol{I} & -\hat{\boldsymbol{Q}}_{LL}^{-1}\boldsymbol{Q}_{LH}^{T} \\ \boldsymbol{0} & \boldsymbol{I} \end{pmatrix} \begin{pmatrix} \hat{\boldsymbol{Q}}_{LL}^{-1} & \boldsymbol{0} \\ \boldsymbol{0} & \hat{\boldsymbol{Q}}_{HH}^{-1} \end{pmatrix} \begin{pmatrix} \boldsymbol{I} & \boldsymbol{0} \\ -\boldsymbol{Q}_{LH}\hat{\boldsymbol{Q}}_{LL}^{-1} & \boldsymbol{I} \end{pmatrix}$$

to precondition our linear system, where the hats $\hat{\boldsymbol{R}}^{-1}$ indicate that the inverses are not explicitly formed; we instead work with approximations to the action of their inverses by using ILU decompositions of $\boldsymbol{R}$. This is because the blocks $\boldsymbol{R}$ are sparse and we do

not wish to store their dense inverses. Thus we have broken down $\boldsymbol{Q}$ so that we can use well-established preconditioning techniques for the linear basis functions on block $\boldsymbol{Q}_{LL}$ together with some choice of preconditioning method for the $\boldsymbol{Q}_{HH}$; we choose ILU decompositions for both of these as they work well for banded matrices [140] (see Section 6.6 for our iteration counts), although other preconditioning choices are possible and further investigation into efficient preconditioning for the $\boldsymbol{Q}_{HH}$ block in particular could be worthwhile.

6.4.2 Preconditioning in 3D

In three dimensions, recall that we instead use the additive Schwarz preconditioner that we discussed in Section 4.6.3. This is in order to give us an evaluation of performance with a different type of preconditioner which has the potential to scale very well in parallel whilst also being very easy to adapt.

6.5 Adaptive Simulation in 1D: Error Norms

6.5.1 Effective Error Control

For initial evaluation of our method, we worked with uniform divisions of a one-dimensional domain of length 2 cm. We performed a 20 ms simulation in 1D with various meshes and error tolerances θ and $\Delta t = 0.01$ ms. The results are plotted in Figure 6.1, which shows that the error control in the adaptive scheme is very effective. We see that with each tolerance, the scheme holds the error at a particular level across a range of fine meshes. The error only starts to rise once the elements become so large that the scheme is already choosing maximal values of p_i in the critical region around the electrical wavefront; in this situation it is no longer able to compensate for the additional error introduced by increasing the element diameter h because no further increase in the p_i-s near the wavefront is possible. As is expected, beyond this point the error follows the quartic convergence

gradient (dashed line) predicted by theory for fixed $p = 4$ FEM [21].

The error was computed using the L^2 norm in time of the H^1 (Sobolev) norm $\|\cdot\|_{1,2,\Omega}$ in space [112], using a non-adaptive p=4 simulation with h=0.0001 cm as a reference solution.

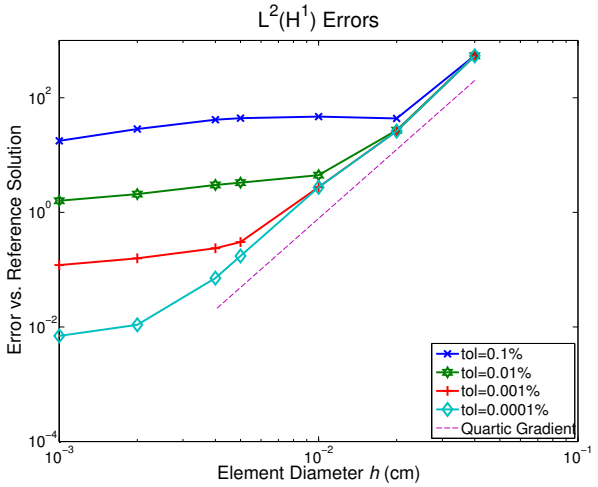


Figure 6.1: Convergence in 1D with various spatial steps h and adaptive tolerances. Each line has seven data points, each corresponding to a different value of h (h=0.001, 0.002, 0.004, 0.005, 0.01, 0.02, 0.04 cm), and each line uses a different error tolerance. Note that with a fixed tolerance (line on the figure), the error is held roughly constant for all values of h below a certain value, dependent on the tolerance. Above that value of h, the error starts to rise because the scheme is using the maximum polynomial basis degree (p=4) available to it, and this is not sufficient to reduce the error to the tolerance level with such large elements. Note that this p=4 limit is simply because we have not coded higher degree basis functions, and also note that when the tolerance is exceeded, the error follows the quartic convergence rate expected for non-adaptive fixed p=4 simulation. The error is measured against high-accuracy solution with h=0.0001 cm and non-adaptive p=4, and all use Δt=0.01 ms.

6.6 Adaptive Simulation in 2D: Activation Times

6.6.1 Meshes and Simulations

In order to evaluate our methods, we worked with a variety of meshes, including differing levels of biological complexity. In two dimensions, we worked with narrow domains $\Omega = [0, 0.25] \times [0, 1]$ cm and $\Omega = [0, 0.25] \times [0, 4]$ cm using a homogeneous isotropic conductivity

tensor field $\sigma(x)$ for basic evaluation of computational performance; the first of these simulates the worst-case scenario for wavefront occupancy in sinus rhythm (i.e. peak occupancy; see Section 7.3.1 for details), and the second represents a case where the wavefront occupancy is lower than this peak, see Section 6.6.2. Because these simulations use stimuli along one of the short edges, they are similar to one-dimensional studies. Therefore, we also perform a simulation using a corner stimulus on a square mesh $\Omega = [0,1] \times [0,1]$ cm in order to create a curved wavefront; these results are in Section 6.6.3.

Including more biological detail, we also use a square mesh $\Omega = [0,1] \times [0,1]$ cm with a rapidly changing cardiac fibre orientation field. While the transmural angular variation of 180° that we use is greater than the 120° seen to occur transmurally with histologically derived fibres [141] (Figure 5.9(a)), this serves to demonstrate that the method is capable of coping with strongly varying fibre orientations. We also use a square domain of the same size with two holes included in order to represent blood vessels passing through the simulation plane, see Figure 5.9(b). For the latter, we used a Laplace-Dirichlet fibre generation method [62]; recall that the fibres describe the direction of fastest conductivity and thus affect the pattern of electrical activation. See Sections 6.6.4 and 6.6.5.

In a final two-dimensional test before moving to three-dimensions, we show snapshots of our scheme working when simulating a re-entrant wave. This was created by first generating a planar wavefront by stimulating along the whole left-hand side of the domain, followed by an early stimulus over the retreating wave-back. The results of this are shown in Figures 6.6 and 6.7.

Where we present timings in this section, they are mean times across the course of the whole simulation. For the non-adaptive comparison cases, they include only the time taken to perform the linear system solve with PCG. For adaptive simulations, they include the time taken to perform the linear system pre-solve, adapt the system matrices and solve the linear system, as in Steps 2, 4 and 5 of the algorithm given in Section 6.3. Note that the time taken to evaluate the error indicator (Step 3) is not included as our simulations are non-parallel; because the error indicator is local to each element, this will

be trivially parallelised and thus cost essentially nothing in terms of compute time. For this reason, we also do not include times for solving for the easily-parallelisable cell model w [139]. We will return to this point in Section 7.4.1. All simulations were performed using MATLAB on a 3.4 GHz CPU, and ran until the wavefront had reached all of the reported test-nodes, unless otherwise stated. We work with a time-step $\Delta t = 0.01$ ms, and all simulations are performed with $\tilde{p} = 4$, whether or not they are adaptive.

We use a conductivity tensor such that the converged conduction velocity parallel to the fibres is roughly 65 cm s^{-1}; this is also the value in all directions in the isotropic case. For the anisotropic simulations, the converged conduction velocity perpendicular to the fibres is about 29 cm s^{-1}, produced by using a conductivity value one-fifth of that used parallel to the fibres. These values are within the physiological range for rabbit hearts [142].

6.6.2 Narrow Meshes with Isotropic Conductivity

We performed simulations on two narrow domains of different lengths with a whole-edge stimulus in order to show the method working in a basic 2D setting. Table 6.1 shows the results for the activation time in the top-right corner of a short narrow domain ($\Omega = [0, 0.25] \times [0, 1]$ cm) stimulated in a thin region along the whole of the lower edge. The table includes simulations in adaptive mode with various error tolerances θ, and also some fixed degree (non-adaptive) simulations for comparison. Note that on each mesh we can achieve the same accuracy with the adaptive scheme as we can with the fixed $p = 4$ simulation at a fraction of the cost (highlighted in red). Note also the superiority over the finest linear simulation in terms of both accuracy and single-iteration linear system solve time. In green we highlight the results which produce around the same accuracy as the finest linear (blue), but with considerably faster linear system solve times; for example with $\theta = 0.3$ on the coarsest mesh we can achieve equivalent accuracy to the finest linear simulation using a linear system which we solve in one-fifth of the time. Figure 6.2 gives an illustrative view of the adaptive selection of element degrees near the wavefront on

this domain; note here the spatially-localised nature of the high spatial gradients and the localised nature of the refinement.

Mean Elt. Diam. (cm)	Simulation Mode	Error Tol. θ (%)	TR Activation Time (ms)	Activ. Time Error (%)	Mean Lin. Sys. Time (s)	Mean PCG Iterations
0.0442	Fixed p=1	N/A	13.18	14.86	0.0010	7.18
	Fixed p=4	N/A	15.47	0.06	0.0326	15.42
	Adaptive	0.7	15.36	0.78	0.0039	10.78
	Adaptive	0.3	15.42	0.39	0.0045	12.34
	Adaptive	0.1	15.47	0.06	0.0056	14.46
0.0221	Fixed p=1	N/A	14.69	5.10	0.0019	5.97
	Fixed p=4	N/A	15.48	0.00	0.0777	9.37
	Adaptive	0.7	15.39	0.58	0.0068	6.89
	Adaptive	0.3	15.45	0.19	0.0079	8.21
	Adaptive	0.1	15.48	0.00	0.0092	8.93
0.0110	Fixed p=1	N/A	15.28	1.29	0.0048	4.43
0.0055	Fixed p=1	N/A	15.43	0.32	0.0222	8.99

Table 6.1: Results for the adaptive scheme on a short narrow test mesh $\Omega = [0, 0.25] \times [0, 1]$ cm. The tolerance used and the activation time in the top-right corner of the domain are given, together with the percentage error in the activation time compared to the converged value and the single-iteration linear system solve CPU time. The mean number of iterations our PCG took to converge on each time-step is shown. Figures 6.3(a) and 6.3(b) show the degrees of freedom used in the adaptive cases compared to the corresponding $p = 1$ simulations, and Figure 6.2 shows a representative polynomial degree distribution for the case $\theta = 0.1$ % on the coarsest mesh reported in this table. We compare the adaptive scheme to the uniform $p = 4$ method on each mesh using red, we highlight the most accurate linear result in blue, and more efficient alternatives to that linear solution in green.

Figure 6.3 shows how the maximum and mean number of degrees of freedom on the short narrow domain $\Omega = [0, 0.25] \times [0, 1]$ change with the tolerance θ. These are the traces with data points marked by crosses, and their scale is on the left of each plot. The values given are ratios of the number of degrees of freedom that a linear-only simulation would use on the particular mesh; note that this changes between plots because the underlying mesh changes. The ratios approach one in both cases; this is particularly interesting for larger values of θ because as can be seen from the diamond-marker error ratio traces also shown in the Figure (right axes scales) and the corresponding data in Table 6.1, even large values of θ provide a huge improvement in accuracy over the linear $p = 1$ case on the same mesh. Note however that for the DOF ratios to reach one, there would have

to be linear elements everywhere over all time; this would cause the error to be equal to that for the $p = 1$ case.

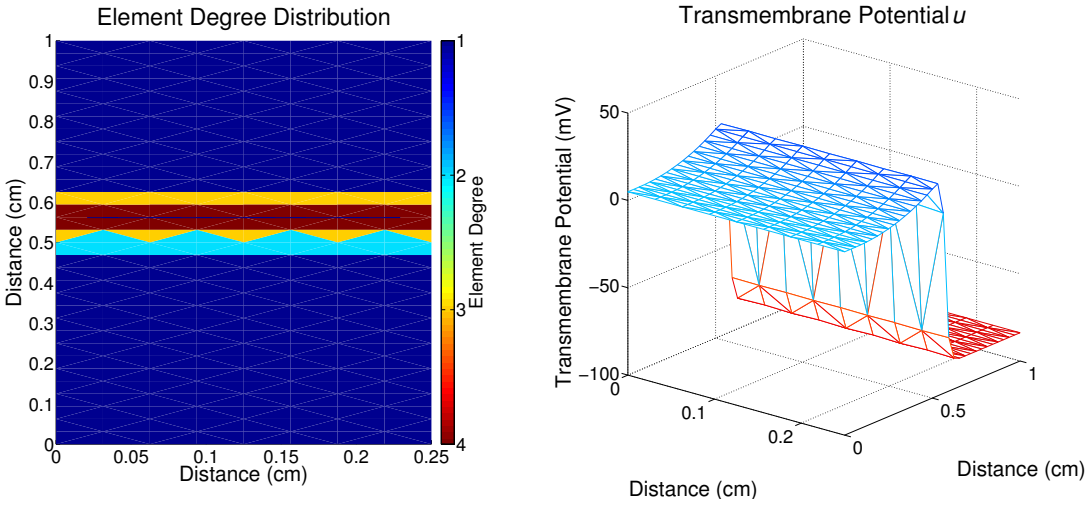
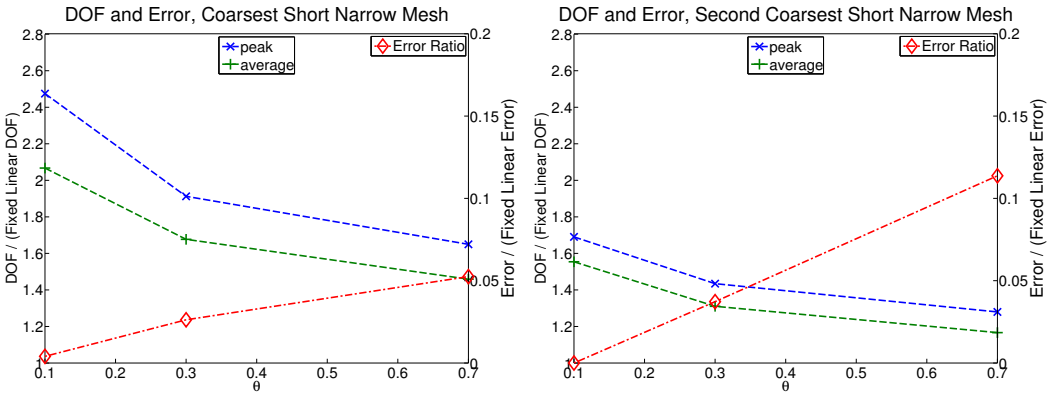


Figure 6.2: Left panel: representative plot of the automatically-chosen element degrees, taken 9 ms after stimulation began. The mesh used is the coarsest short narrow mesh (mean element diameter 0.0442 cm), and the tolerance used was $\theta = 0.1$ %. The wavefront (contour u=0) is shown by the blue line. Right panel: plot of the transmembrane potential u showing that the steep behaviour occurs only near the wavefront. The colouring shows the same data as the height of the surface in the z-direction. Physiologically, $u(x, t)$ remains well within the region [-100,50] mV. See Table 6.1 for full data.

In Table 6.2, we show results for a long narrow domain. These simulations differ from those on the short narrow domain only in that we use meshes of $\Omega = [0, 0.25] \times [0, 4]$ cm. This gives us an idea of the performance of the scheme when the wavefront occupies a smaller percentage of the domain. Here, we see even more impressive analogues of the results for the short narrow mesh; the adaptive scheme delivers considerable performance improvements for the same error over fixed $p = 4$ simulations (red cells), and it is easily superior on measures of both speed and accuracy to the finest fixed $p = 1$ simulation (blue vs. green cells). For example, with $\theta = 0.1$, simulation on the coarsest mesh generates the approximation nine times faster than the finest linear can, and with a large increase in accuracy.



(a) Short narrow domain, coarsest mesh (0.0442 cm mean element size). See Table 6.1 for more data.

(b) Short narrow domain, second coarsest mesh (0.0221 cm mean element size). See Table 6.1 for more data.

Figure 6.3: Plots showing the average (mean) and the peak number of degrees of freedom used over the course of the adaptive simulations shown in Table 6.1 (scale on the left axes) and the error in each case (right axes). The DOF values are ratios of the number of degrees of freedom that a fixed linear simulation on the same mesh would use, and the errors are ratios of the percentage activation time errors at a single point in the domain (error using given value of θ) / (error using fixed linears on the same mesh). Each plot shows how the number of degrees of freedom used changes with the tolerance θ used.

Mean Elt. Diam. (cm)	Simulation Mode	Error Tol. θ (%)	TR Activation Time (ms)	Activ. Time Error (%)	Mean Lin. Sys. Time (s)	Mean PCG Iterations
0.0442	Fixed p=1	N/A	52.42	15.38	0.0020	6.94
	Fixed p=4	N/A	61.92	0.05	0.0998	13.17
	Adaptive	0.7	61.41	0.87	0.0078	9.82
	Adaptive	0.3	61.68	0.44	0.0085	11.14
	Adaptive	0.1	61.92	0.05	0.0098	12.95
0.0221	Fixed p=1	N/A	58.67	5.29	0.0050	5.35
	Fixed p=4	N/A	61.95	0.00	0.2781	8.30
	Adaptive	0.7	61.56	0.63	0.0200	6.26
	Adaptive	0.3	61.84	0.18	0.0219	7.34
	Adaptive	0.1	61.94	0.01	0.0226	7.92
0.0110	Fixed p=1	N/A	61.12	1.34	0.0172	4.10
0.0055	Fixed p=1	N/A	61.75	0.32	0.0917	8.22

Table 6.2: Results for the adaptive scheme on a long narrow test mesh $\Omega = [0, 0.25] \times [0, 4]$ cm. The tolerance used and the activation time in the top-right corner of the domain are given, together with the percentage error in the activation time compared to the converged value and the single-iteration linear system solve CPU time. The mean number of iterations our PCG took to converge on each time-step is shown. We compare the adaptive scheme to the uniform $p = 4$ method on each mesh using red, we highlight the most accurate linear result in blue, and more efficient alternatives to that linear solution in green.

6.6.3 Square Mesh with Isotropic Conductivity

We tested our method with homogeneous isotropic conductivity tensor σ by stimulating in the the lower-left corner of a 1 cm $\times$ 1 cm domain to obtain a curved propagating wavefront. The activation time percentage errors for the node in the top-right (diagonally opposite the stimulus) corner are given in Table 6.3. Note that we can achieve very similar accuracy to the finest linear solution in one-seventh of the linear system time by using θ=0.3 % on the coarsest mesh (highlighted in red), or we can obtain a huge increase in accuracy (greater than fifty times more accurate, to convergence at this measurement resolution) with an almost three-times faster linear system solve by using θ=0.1 % on the second coarsest mesh. The results which show a clear efficiency improvement are shown in green. Figure 6.4 shows how the high-order basis functions are localised to the steep wavefront on the second coarsest mesh with θ=0.5 %. Note that with this large value of θ, the scheme has only deemed the use of a small number of degree-four elements to be necessary.

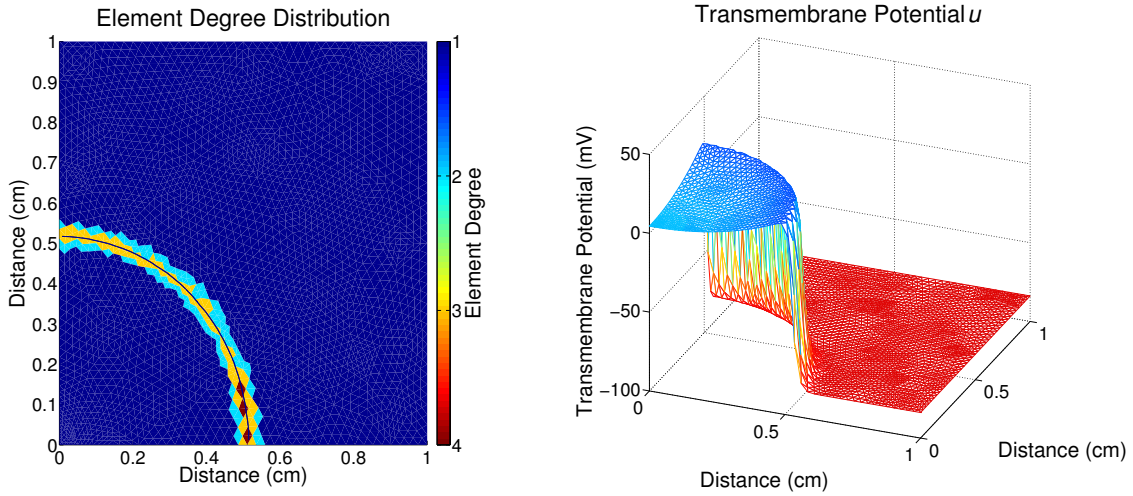


Figure 6.4: Left panel: representative plot of the automatically-chosen element degrees, taken 9 ms after stimulation began. The mesh used is the second coarsest square mesh without holes (mean element diameter 0.0226 cm), and the tolerance used was θ = 0.5 %. The wavefront (contour u=0) is shown by the blue line. Right panel: plot of the transmembrane potential u showing that the steep behaviour occurs only near the wavefront; the colouring shows the same data as the height of the surface in the z-direction. See Table 6.3 for the full data.

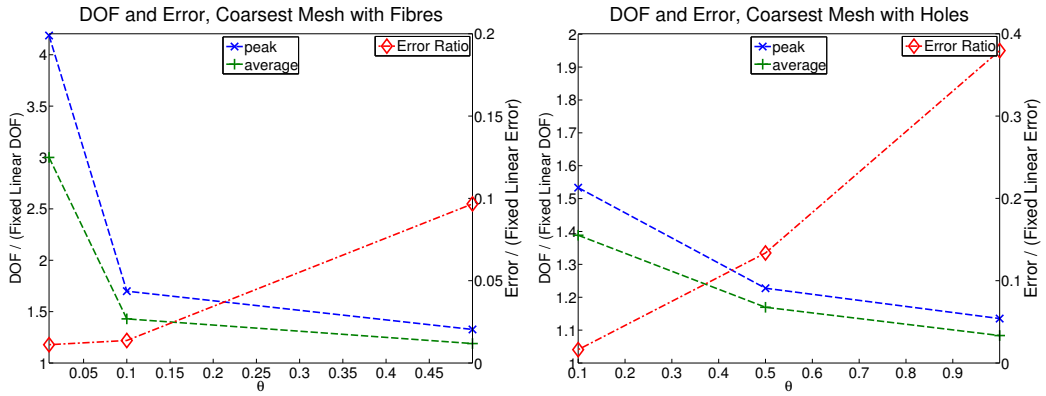
Mean Elt. Diam. (cm)	Simulation Mode	Err. Tol. θ (%)	TR Activation Time (ms)	Activ. Time Error (%)	Mean Lin. Sys. Time (s)	vs. Finest Lin.: Error	vs. Finest Lin.: Speed-up
0.0452	Fixed p=1	N/A	18.33	18.93	0.0016	35.7	44.5
	Fixed p=4	N/A	22.56	0.22	0.0822	0.4	0.9
	Adaptive	0.5	22.32	1.28	0.0088	2.4	8.1
	Adaptive	0.3	22.45	0.71	0.0101	1.3	7.0
	Adaptive	0.1	22.57	0.18	0.0136	0.3	5.2
0.0226	Fixed p=1	N/A	20.92	7.47	0.0035	14.1	20.3
	Fixed p=4	N/A	22.61	0.00	0.2025	0.0	0.4
	Adaptive	0.5	22.49	0.53	0.0209	1.0	3.4
	Adaptive	0.3	22.58	0.13	0.0223	0.3	3.2
	Adaptive	0.1	22.61	0.00	0.0259	0.0	2.7
0.0113	Fixed p=1	N/A	22.12	2.17	0.0113	4.1	6.3
0.0057	Fixed p=1	N/A	22.49	0.53	0.0712	1	1

Table 6.3: Results for the adaptive scheme on a square domain $\Omega = [0,1] \times [0,1]$ cm, stimulated in the lower-left corner in order to produce a curved wavefront. The tolerance used and the activation time in the top-right corner of the domain are given, together with the percentage error in the activation time compared to the converged value and the single-iteration linear system solve CPU time. The final two columns present the percentage activation time error as a fraction of the finest linear error, and the speed-up fraction (finest p=1 linear system time) / (linear system time). We highlight the simulations showing clear efficiency improvement in green, and one which would also make a good choice in red.

6.6.4 Square Mesh with Anisotropic Conductivity

Representing the case of rapidly changing cardiac fibre orientation using the vector field shown in Figure 5.9(a) and using activation times at the four points shown there, we evaluated our adaptive scheme on a square domain $\Omega = [0,1] \times [0,1]$ cm and with a suitable σ. The results of this are given in Table 6.4; notice that the error remains quite large with $p = 1$ (blue cells), even on very fine meshes, whereas the adaptive scheme can produce very much higher quality approximations. It does so efficiently; for example, with $\theta = 0.1$ %, we cut the error to under one-tenth of that of the finest $p = 1$ simulation, obtaining the approximation using a linear system which can be solved twice as fast, or we can obtain a similar worst-node error in around one-third of the time (green cells).

Figure 6.5(a) presents the DOF and error ratios for the simulation data. See Section 6.6.2 for a description of these plots, and note that between Figures 6.5(a) and 6.5(b), the scales on both the x- and y-axes differ.



(a) Square domain with fibres, coarsest mesh (0.0226 cm mean element size). See Table 6.4 for more data.

(b) Square domain with holes, coarsest mesh (0.0296 cm maximum element size). See Table 6.5 for more data.

Figure 6.5: Plots showing the average (mean) and the peak number of degrees of freedom used over the course of the adaptive simulations shown in Tables 6.4 and 6.5 (scales on the left axes) and the error in each case (right axes). The DOF values are ratios of the number of degrees of freedom that a fixed linear simulation on the same mesh would use, and the errors are ratios of the percentage activation time errors at a single point in the domain (error using given value of θ) / (error using fixed linears on the same mesh). Each plot shows how the number of degrees of freedom used changes with the tolerance θ used.

Mean Elt. Diam. (cm)	Simulation Mode	Error Tol. θ (%)	Node 1 Error (%)	Node 2 Error (%)	Node 3 Error (%)	Node 4 Error (%)	Lin. Sys. Time (s)
0.0226	Fixed p=1	N/A	23.38	18.64	7.40	19.13	0.0036
	Fixed p=4	N/A	0.26	0.14	0.00	0.09	0.2841
	Adaptive	0.5	3.05	2.26	0.75	2.53	0.0227
	Adaptive	0.1	0.32	0.20	0.00	0.18	0.0280
	Adaptive	0.01	0.26	0.14	0.00	0.09	0.0586
0.0113	Fixed p=1	N/A	10.65	7.94	2.30	8.12	0.0124
0.0055	Fixed p=1	N/A	3.67	2.58	0.62	2.62	0.0649

Table 6.4: Results for the adaptive scheme on a square test mesh $\Omega = [0, 1] \times [0, 1]$ cm with fibres. The tolerance used and the percentage error in the activation time compared to the converged value are given, as is the mean single-iteration linear system solve CPU time. The best linear solution is highlighted in blue, and efficiency improvements over it are shown in green. The activation times were measured at the four points marked in Figure 5.9(a); the domain and stimulus site used are shown there also. Figure 6.5(a) shows the degrees of freedom used in the adaptive cases compared to the corresponding $p = 1$ simulation.

6.6.5 Square Domain with Holes and Anisotropic Conductivity

Using the domain, Laplace-Dirichlet-derived fibres and three test nodes shown in Figure 5.9(b), we can investigate how adaptivity performs when holes representing blood vessels

passing through the domain are present. Table 6.5 shows the results; note that the error in the solution with $p = 1$ on a very fine mesh (blue cells) is even worse than it is in the case of the square-with-fibres simulations on the domain shown in Figure 5.9(a) (data in Table 6.4). Here we see that with $\theta = 0.1$ % (shown in green) we can reduce the error to one-tenth of the best linear approximation with a linear system which can be solved in half the time. Another clear improvement is shown in red.

In addition to the superiority to piecewise-linear FEM, the same observations that can be made in the previous cases hold true here too; specifically, the adaptive scheme allows us to achieve equivalent accuracy as the fixed $p = 4$ approach, but at a fraction of the cost.

Max. Elt. Diam. (cm)	Simulation Mode	Error Tol. θ (%)	Node 1 Error (%)	Node 2 Error (%)	Node 3 Error (%)	Lin. Sys. Time (s)	Err. Ratio vs. Best Linear
0.0296	Fixed p=1	N/A	11.17	18.85	24.61	0.0075	6.02
	Fixed p=4	N/A	0.12	0.20	0.35	0.4827	0.09
	Adaptive	1	5.64	7.49	9.34	0.0297	2.42
	Adaptive	0.5	1.96	2.77	3.29	0.0343	0.84
	Adaptive	0.1	0.25	0.30	0.40	0.0425	0.11
0.0148	Fixed p=1	N/A	5.40	8.95	11.47	0.0186	2.80
0.0076	Fixed p=1	N/A	2.33	3.42	4.09	0.0952	1 (by def.)

Table 6.5: Results for the adaptive scheme on a square test mesh $\Omega = [0, 1] \times [0, 1]$ cm with two holes and Laplace-Dirichlet fibres. In each case, we show the worst value over the three sampled nodes of (percentage error)/(finest linear percentage error). Note that the meshes are identified by maximum element diameter rather than mean here due to the many small elements near the holes. The tolerance used and the percentage error in the activation time compared to the converged value are given, as is the mean single-iteration linear system solve CPU time. The activation time measurements were made at the three points marked in Figure 5.9(b); the domain and stimulus site used are shown there also. Figure 6.5(a) shows the degrees of freedom used in the adaptive cases compared to the corresponding $p = 1$ simulation. The best linear solution is shown in blue, and significant efficiency improvements over it are shown in red and in green.

Figure 6.5(b) presents the DOF and error ratios for the simulation data in Table 6.5. See Section 6.6.2 for a description of these plots, and note that between Figures 6.5(a) and 6.5(b), the scales on both the x- and y-axes differ.

6.6.6 Re-entrant Behaviour

Of particular interest is the situation where the normal, ordered electrical propagation breaks down due to tissue damage or external influence such as electric shock or physical impact [143]. In this case, the waves of electrical activation can re-enter regions which were recently activated, causing dangerous cardiac arrhythmias. This can be induced on a sufficiently large domain by stimulating along one tissue edge to create a planar wave and waiting for the full action potential to pass through, and then applying a second early stimulus to a region containing both excitable and refractory tissue near the wave-back. The border of this early stimulus can be seen in Figure 6.6. Four time instants showing the induced wave are given in Figures 6.6 and 6.7. The right-hand panel in each pair shows the transmembrane potential distribution, and the left shows the elements coloured according to the polynomial degree that the adaptive software automatically selected for each. These results serve to give a feel for how the scheme would perform for a class of simulations different to the simple pacing cases presented above, demonstrating that the adaptive scheme refines p as expected in this more complicated case. We did not perform convergence studies for this simulation because the scale of high-accuracy (i.e. converged), fixed polynomial degree simulations on the large domain size necessary to support re-entrant waves becomes problematic for the non-parallel software we are using.

6.7 Adaptive Simulation in 3D: Activation Times

6.7.1 Small Test Cube

We return to the cube mesh investigated in Section 5.5.1, examining the efficiency that we gain by using the adaptive algorithm. Recall that the cube has dimensions $0.2 \times 0.2 \times 0.2$ cm and that we use Niederer benchmark conductivities [58]. We record the activation time of the last node in the domain, in the corner of the cube furthest from the stimulus site; see Figure 5.11 in which this measurement is taken at the corner marked with an

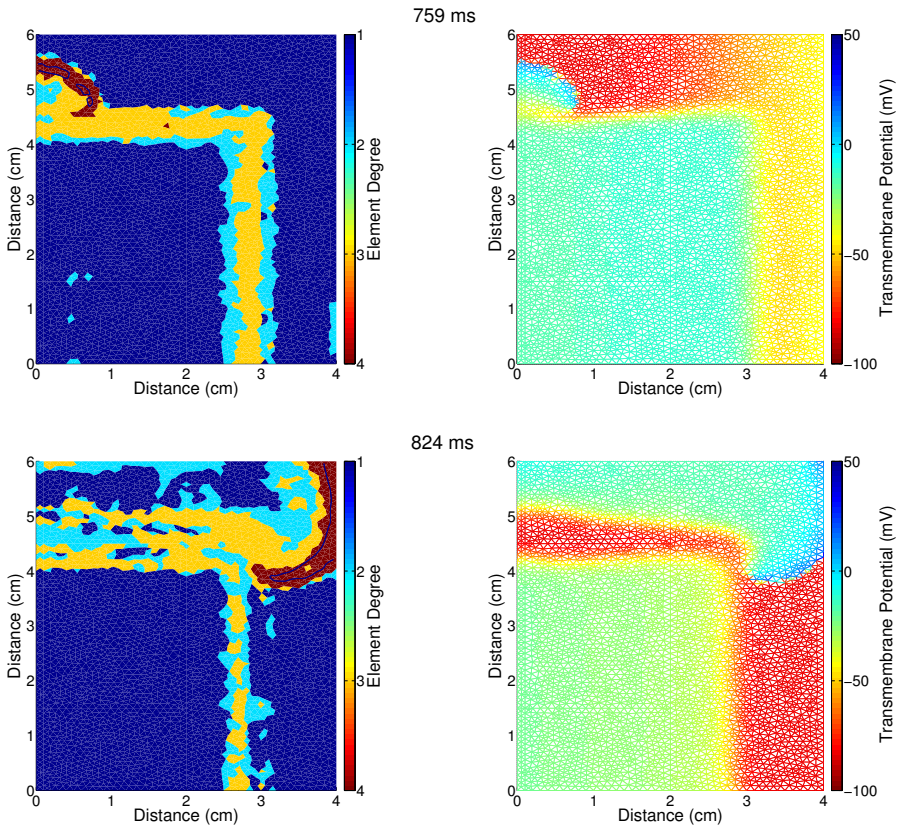


Figure 6.6: Four time instants in a simulation of re-entrant behaviour. The right-hand panel of each pair shows the transmembrane potential u distribution in the domain, and on the left we have the mesh elements coloured according to the polynomial degree that the adaptive scheme has automatically chosen for each. The contour $u = 0$ is drawn on the left panels; note that sometimes it appears as a broken line as it is calculated using the linear basis component of the solution only, which has insufficient resolution to correctly identify the contour at some points in time. Plots continue in Figure 6.7.

eight.

We present the results of this in Table 6.6. As we have come to expect, given our lower dimensional test cases, we see that the errors are controlled effectively by the adaptive scheme and that it shows improved efficiency. Notice that the additive Schwarz preconditioner (see Section 4.6.3) keeps the number of iterations that are required for convergence of the conjugate gradient algorithm small. Of particular interest is the result that we can achieve roughly the same error by using $\theta = 20\%$ on the coarsest grid as we can on the finest grid with $p = 1$ using a linear system which can be solved four times faster. Compared to the same linear solution, also of interest is the fact that we

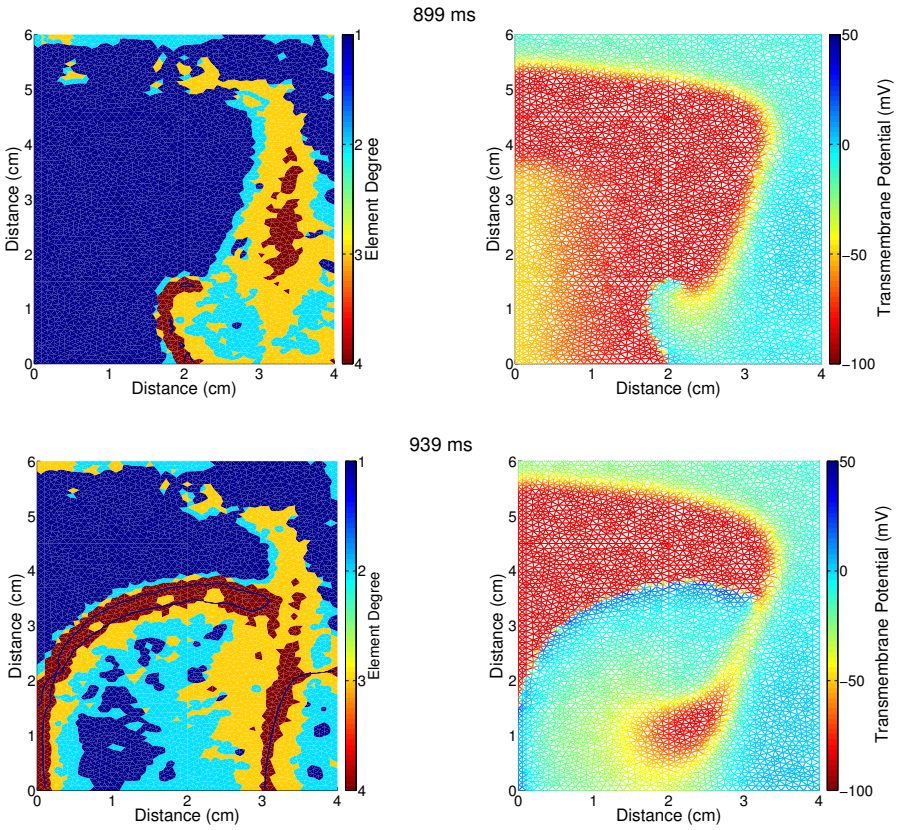


Figure 6.7: Continuation of Figure 6.6.

can achieve a sixteen-times smaller percentage error in about the same amount of CPU time by taking $\theta = 1\%$. While these results are promising, it needs to be remembered that because the simulation domain is tiny, the proportion of it which will be occupied by the wavefront at any one time is very large. This means that adaptivity, which typically refines in p at the wavefront, is not expected to come close to its full potential on this test problem. Thus, the accuracy is of more interest than CPU times in this case. Note that we use $\tilde{p} = 4$ in all simulations, which can result in values for the mean iteration time which is much larger than the PCG time (which is our main interest here), as seen in the $h = 0.01$ cm case in Table 6.6. Such a high value of $\tilde{p}$ may not be necessary when p is small.

Figure 6.9 gives a graphical presentation of the percentage errors for the adaptive simulations given in Table 6.6, showing how the numerical results respond to the target error parameter θ. This figure provides further evidence that the error control scheme

is highly effective in practice; the error response is approximately linear below 20%, and above this value of θ it starts to drop off. Note that for all values of θ, the error remains below the line $y = x$, meaning that for this simulation it happens that θ is an upper bound for the error. While these linearity and bounding properties say nothing about the general case, they provide a useful feel for the sort of behaviour we might expect, and a complement to the impressive error norm traces given in Figure 6.1. We note however that the precise form that Figure 6.9 takes will likely depend on many factors, such as conductivities, fibre orientation, domain geometry and mesh resolution.

Elt. Diam. h (cm)	Simulation Mode	Final Activ. Time (ms)	% Activ. Time Error	PCG Time (s)	Mean It. Time (s)	Mean PCG Iterations	Max. PCG Iterations
0.04	p=1	8.32	31.4	0.05	0.15	7.3	9
	p=2	8.93	26.4	0.12	0.22	12.3	16
	p=3	12.77	5.3	0.39	0.49	13.3	16
	p=4	12.19	0.5	1.57	1.68	14.3	17
	$\theta = 50\%$	8.73	28.0	0.22	0.34	12.2	18
	$\theta = 40\%$	9.29	23.4	0.25	0.37	12.6	18
	$\theta = 30\%$	9.61	20.8	0.30	0.42	13.1	19
	$\theta = 20\%$	10.27	15.3	0.38	0.50	13.6	19
	$\theta = 15\%$	10.63	12.4	0.46	0.58	13.8	19
	$\theta = 12.5\%$	10.93	9.9	0.48	0.60	13.8	19
	$\theta = 10\%$	11.10	8.5	0.54	0.66	13.9	19
	$\theta = 5\%$	11.93	1.6	0.65	0.77	13.9	19
	$\theta = 1\%$	12.25	1.0	1.29	1.40	14.2	18
0.02	p=1	8.58	29.3	0.23	0.52	6.7	9
	p=2	10.68	12.0	0.76	1.06	12.1	14
	p=3	12.18	0.4	2.90	3.21	12.4	15
	p=4	12.07	0.5	12.61	12.93	13.4	16
0.01	p=1	10.11	16.7	1.39	3.19	6.0	8

Table 6.6: Activation times of the final node in the $0.2 \times 0.2 \times 0.2$ cm small cube domain shown in Figure 5.11 using various h and p values. The reference solution, generated using h=0.01 and p=4, gave an activation time of 12.13 ms. The simulation mode is given in terms of either p or θ, for the fixed polynomial degree or the adaptive scheme error tolerance, respectively. Fibres and conductivities are as given in [58]. The timings presented are for the linear system only (additive Schwarz PCG) and for the whole-timestep (additive Schwarz PCG + cell updates, etc.). The mean number of iterations that the PCG took to converge are given. The degrees of freedom for the meshes used was presented in Table 5.8, and a plot of the wavefront at 6 ms is given in Figure 6.8. The response of the percentage error to the error tolerance parameter θ is also presented graphically in Figure 6.9

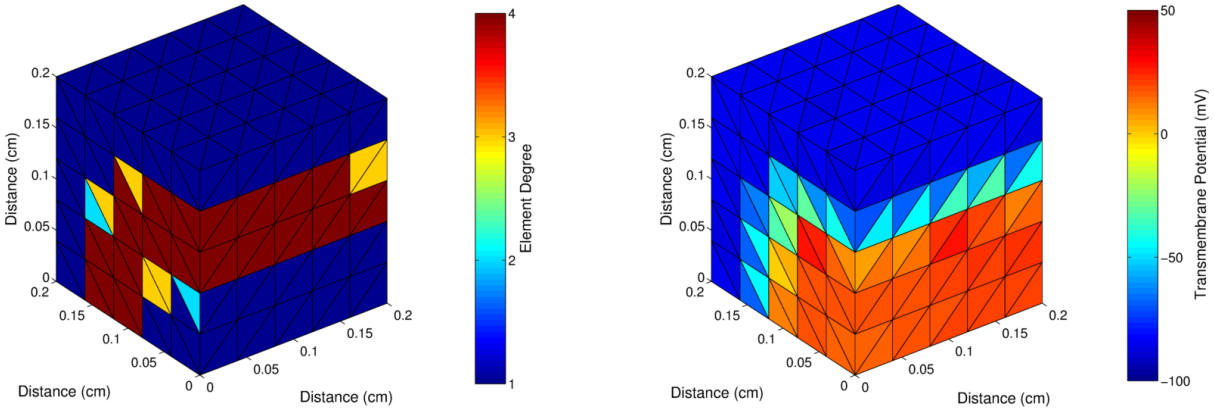


Figure 6.8: The adaptive scheme applied to the cube benchmark problem using a target error parameter $\theta = 10\%$ and $h = 0.04$ cm. Element degrees (left) and transmembrane potential (right) at 6 ms. Notice the narrow band of high-order elements tracking the transmembrane potential wavefront.

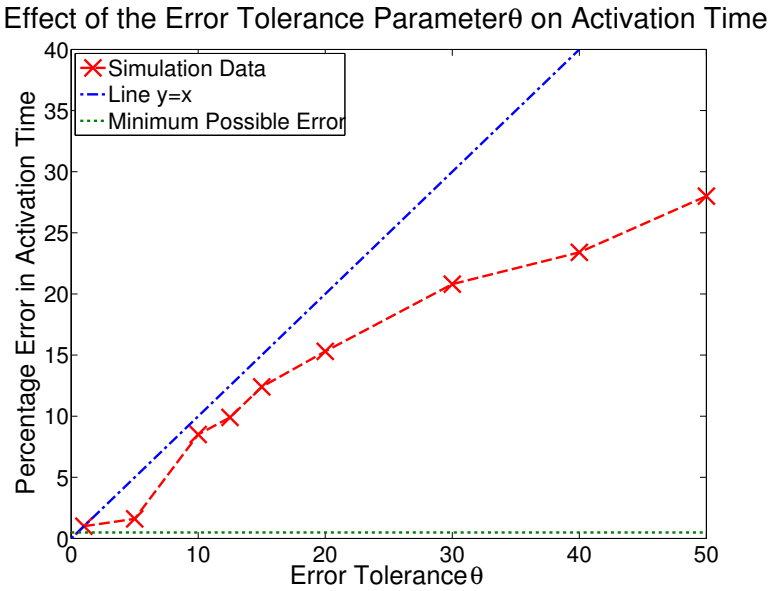


Figure 6.9: The effect of varying θ on the percentage activation time error for the final node in the domain on the 3D cube test mesh at h=0.04 cm. Data is for the activation time at Node 8; see Table 6.6 and Figure 5.11. Note the approximately linear trend for small values of θ, and that this breaks down for small values of θ when we approach the minimum error possible on this mesh (i.e. that of a fixed $p = 4$ simulation).

6.7.2 The Niederer Benchmark

We now return to the Niederer et al. 3D test problem that we examined in Section 5.5.2 and evaluate the performance of p-adaptivity for its solution. Recall that this uses a

simulation domain of size to be $0.3 \times 0.7 \times 2$ cm at mesh discretisations of 0.05, 0.02 and 0.01 cm, and uses anisotropic conductivity. The domain was presented in Figure 5.12, and further details were given in Section 5.5.2. Figure 6.10 shows the element degree distribution and transmembrane potential of a simulation using $\theta = 10\%$, at 18 ms.

The results of solving this test problem with our adaptive technique are presented in Table 6.7. We start to see the full power of adaptivity in 3D in this test, with the adaptive scheme producing the same accuracy as the finest linear solution at node eight in less than one-tenth of the time. Notice that we achieve comparable accuracy at node eight to this finest linear case when we use $\theta = 15\%$; this is similar to the $\theta = 20\%$ value that we saw on the small cube mesh in Section 6.7.1. This indicates that we have a consistency in the response of the numerical scheme to differing values of θ between similar simulations. Note also that we can obtain good accuracy at all nine test-nodes using $\theta = 5\%$, obtaining in this case a seven-fold faster linear system than that of the finest linear.

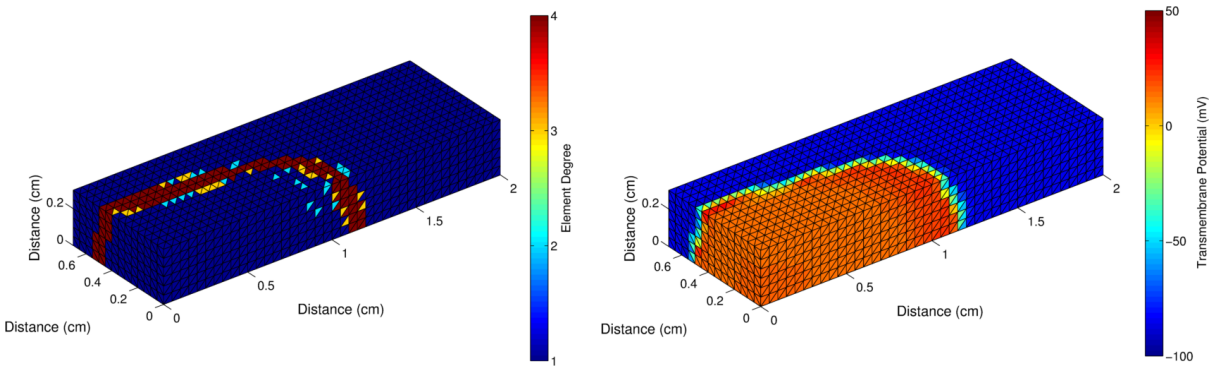


Figure 6.10: The adaptive scheme applied to the Niederer benchmark problem using a target error parameter $\theta = 10\%$ and $h = 0.05$ cm. Element degrees (left) and transmembrane potential (right) at 18 ms. Notice the narrow band of high-order elements tracking the transmembrane potential wavefront.

Figure 6.11 shows a similar result for the Niederer problem as Figure 6.9 did for the small cube. We see a linear response to θ in the range $[5, 20]$, which breaks down for larger values of θ and as we approach the minimum possible error on this mesh, which corresponds to a simulation with $p = 4$ set in all the elements. As we commented for

h	Sim. Mode	Activation Time in ms at Node: 1	2	3	4	5	6	7	8	9	PCG+Adj. Time (s)	Full It. Time (s)
0.05	p=1	1.25	27.95	5.87	27.12	15.96	29.24	14.40	33.22	13.40	0.63	1.48
	p=2	1.24	30.08	6.29	30.03	19.45	33.49	19.15	32.70	15.86	2.50	3.33
	p=3	1.24	31.52	7.78	32.85	23.10	38.24	26.33	40.80	18.66	9.54	10.37
	p=4	1.24	31.90	7.54	33.66	25.21	40.02	28.89	41.52	19.24	42.85	43.68
	θ=40%	1.24	29.54	6.72	30.15	21.39	34.68	20.97	34.56	16.50	3.42	4.34
	θ=20%	1.23	30.76	7.38	31.11	23.22	36.50	24.68	37.04	17.60	4.32	5.25
	θ=15%	1.24	31.04	7.62	31.54	23.94	37.31	25.37	38.35	18.19	4.72	5.64
	θ=12.5%	1.22	31.10	7.57	31.88	24.32	37.86	26.13	39.07	18.47	5.20	6.41
	θ=10%	1.21	31.18	7.80	32.21	24.59	38.31	26.84	39.65	18.72	5.24	6.17
	θ=5%	1.24	31.76	7.69	33.05	25.44	39.43	28.44	41.01	19.41	6.89	8.17
	θ=1%	1.24	31.98	7.50	33.57	25.50	40.03	29.07	41.57	19.26	13.47	14.40
0.02	p=1	1.23	30.64	7.12	30.47	21.65	33.56	21.15	32.73	15.89	-	-
	p=2	1.24	31.92	7.81	32.69	24.30	39.16	25.11	39.47	18.61	-	-
	p=3	1.24	32.19	8.46	33.41	26.65	41.84	28.97	43.01	19.93	-	-
	p=4	1.25	32.23	8.37	33.52	26.64	42.16	28.78	43.20	20.06	-	-
0.01	p=1*	1.23	31.46	7.82	31.93	25.44	38.04	26.02	38.17	17.87	51.58	51.82

Table 6.7: 3D activation times on the $0.3 \times 0.7 \times 2$ cm Niederer test problem domain shown in Figure 5.12 using various h (cm) and p values. The timings presented are for the additive Schwarz PCG-solve only, and for the whole-timestep full iteration (PCG, cell updates, etc.). The wavefront at 18 ms is shown in Figure 6.10, the degrees of freedom for the meshes used was presented in Table 5.8, and the response of the percentage error to the error tolerance parameter θ is presented in Figure 6.11. Asterisked simulations were run with $\tilde{p} = 1$, and some timings are absent because the simulation was run on a different computer, due to memory restrictions on the primary test machine.

the cube domain in Section 6.7.1, this reinforces the idea that the error control behaves in an intuitive way as we adjust θ. This property is desirable for the software user, but we reiterate that we can not infer anything about the general case from these activation time error response results.

6.8 Whole Rabbit Heart Mesh: Simulations and Predictions

In this section, we compute an estimate for the theoretical maximum speed-up for generating a solution with an accuracy that would be suitable for some research applications (specifically, with a target error of 5%) on a complete mesh of the ventricles in a rabbit heart and using adaptivity. Comparisons are made using systems preconditioned by the

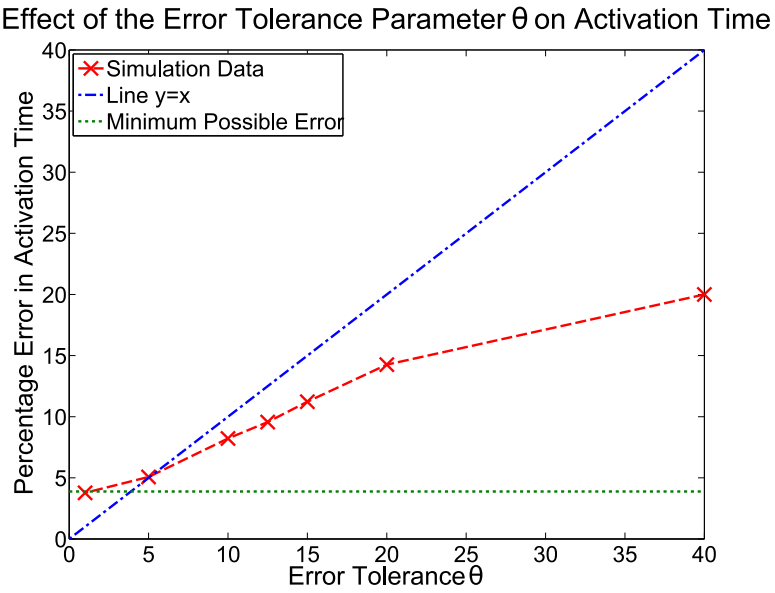
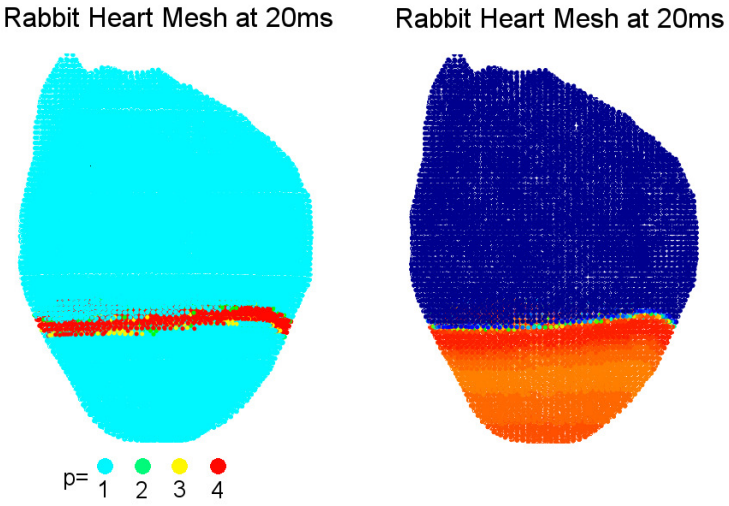


Figure 6.11: The effect of varying theta on the percentage activation time error for the final node in the domain on the Niederer test problem mesh at h=0.05 cm. Data is for the activation time at Node 8; see Table 6.7 and Figure 5.12. The solution $h = 0.02$, $p = 4$ is taken as the reference solution. Note the approximately linear trend for small values of θ, and that this breaks down for small values of θ when we approach the minimum error possible on this mesh (i.e. that of a fixed $p = 4$ simulation).

additive Schwarz method described in Section 4.6.3. In single-thread simulation faster preconditioning methods are available, but because we wish to consider a method which will work well in parallel, we select additive Schwarz as this class of methods has been found to be "optimal and scalable", and to provide "the best CPU times to date" in parallel studies [101, 102]. Direct methods such as Gaussian elimination are unlikely to be competative on such large meshes, even for the linear presolve. Note that the presolve represents only a very small fraction (1% in our case) of the total solution work, as we shall see.

We converted the data structures of the magnetic-resonance-derived, anatomically-accurate rabbit heart mesh of Bishop et al. [95] into our high-order format and simulated propagation using isotropic conductivities and an apical stimulus. We set an error tolerance parameter of $\theta = 5\%$; the resulting propagation and polynomial degree distribution is shown at 20 ms in Figure 6.12, and at 56 ms in Figure 6.13. Note that we were unable

to perform any convergence analysis on this mesh due to the huge computational demand that would be involved in working with a refined mesh in our single-threaded software. However, the mesh has mean element diameter of approximately 0.06 cm, and the conductivities are all set to one, so with $\theta = 5\%$ on the rabbit mesh we can expect similar accuracy to the $\theta = 5\%$ adaptive Niederer-problem studies performed in this work. This mesh contains 431,990 tetrahedra, and 82,619 degrees of freedom at $p = 1$ and 4,771,084 at uniform $p = 4$. Due to the adaptivity we shall never use more than a small fraction of the $p = 4$ degrees of freedom at the same time.

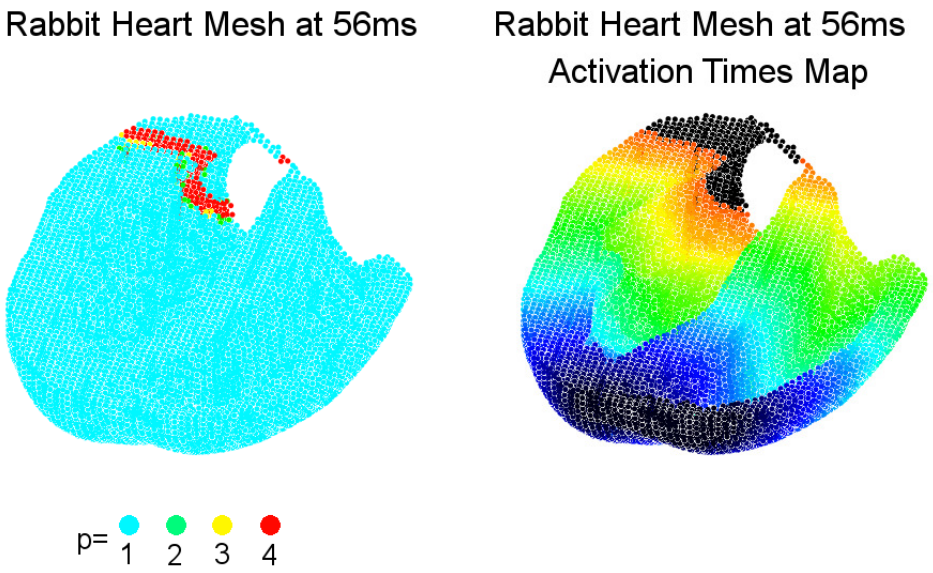


(a) Local polynomial degree.

(b) Transmembrane potential. The dark blue tissue is at the resting potential.

Figure 6.12: The adaptive scheme applied to an anatomically-accurate rabbit heart mesh and using a target error parameter $\theta = 5\%$. Notice the narrow band of high-order elements tracking the transmembrane potential wavefront.

We can compute some estimates of the sort of speed that we might expect to obtain with this mesh with $\theta = 5\%$ if we were using a high-performance computing facility. Ignoring the preconditioner to begin with, we can consider the number of floating point operations (FLOPS) that are required in order to perform a matrix-vector multiplication using the system matrix, as this constitutes the majority of the cost of the conjugate gradients algorithm, after preconditioning. Comparisons of the number of floating point



(a) Local polynomial degree.

(b) Activation Times. The black region at the top of the figure has not yet been activated.

Figure 6.13: Cut-away view of the anatomically-accurate rabbit heart mesh after 56 ms of adaptive simulation using a target error parameter $\theta = 5\%$. Notice the narrow band of high-order elements tracking the transmembrane potential wavefront in Figure 6.13(a), located at the red region of most recently activated tissue shown in Figure 6.13(b).

operations required for this will then provide an indication of the speed we might expect on a powerful supercomputer.

As an example, we take the data for the rabbit heart after 20 ms of simulation. This is the state shown in Figure 6.12. We then perform the estimation

$$\frac{\text{number of nonzeros in adapted linear system matrix}}{\text{number of nonzeros in 0.012 cm } p\text{=1 linear system matrix}} \approx 0.07,$$

where we have chosen the fivefold finer mesh with h=0.012 cm as one that we could expect, in the light of the results in Table 6.7, to provide similar accuracy to the $\theta = 5\%$ solution generated on the coarse mesh. This estimate reveals that a matrix-vector multiply is around 7% as costly for the adapted system as for a fine $p = 1$ system. All that remains to be accounted for is the number of matrix-vector multiplications that are required in each

case; this information is available via the iteration counts we have recorded when using the Schwarz preconditioner. Referring to Table 6.6, we estimate that the ratio of required iterations for the adapted coarse to the fine p=1 linear systems would be $I := 19/8$, which would result in an overall floating point ratio of around 17%, representing an overall sixfold speedup.

However, this calculation, which applies when the wavefront is present in the domain, does not tell the whole story. For a study of sinus rhythm on the rabbit heart, the wavefront is only present in the domain for around 60 ms of the approximately 300 ms full cycle length, so for around $T_{NW} := 240/300$ of the time, the adapted linear system would have no high-order elements present, making the adapted linear system around 1% of the size of the fine system in terms of its number of non-zero matrix entries. In this case, we will not have the factor 19/8. We can also include the cost of performing the linear presolve on each iteration, which will just be another 1%. Thus the full-cycle cost with adaptivity can be approximated as

$$\frac{240}{300} \times 1\% + \frac{60}{300} \times 17\% + 1\% \approx 5\%$$

of that of the fine linear. We note that this final point is not always relevant because it is not always necessary to run simulations for whole cycles, but there are cases where this is important, such as when investigating slow APD adaptation [144, 145]. This can require running for hundreds of cardiac cycles, and is currently limited in 3D due to the computational cost.

We can perform a similar calculation for the cost of the Schwarz preconditioner as follows. The preconditioner breaks down into a local matrix problem of size $S \times S := 15 \times 15$ for linear degrees of freedom. This size increases with p; at $p = 4$ it is approximately $B \times B := 300 \times 300$. We store these problems as an LU decomposition, and the number of FLOPS required to perform back-substitution using a triangular matrix of size $n \times n$ is n^2. There are two triangular factors, so this cost is doubled. At the 20 ms state shown in

Figure 6.12, approximately 2.5% ($F_H := 0.025$) of the elements have degree greater than one; we assume that this is representative and that all elements have degree either one or four. Let P be the number of local matrix problems that we must solve. We estimate the FLOPS required for the preconditioner over a whole cardiac cycle using the expression

$$\left[T_{NW} P(2S^2)\right] + \left[(1 - T_{NW}) IP \times \left((2 \times B^2) F_H + (2S^2)(1 - F_H)\right)\right]$$

as

$$\left[\frac{240}{300} \times P \times 2 \times 15^2\right] + \left[\frac{60}{300} \times \frac{19}{8} \times P \times \left(2 \times 300^2 \times 0.025 + 2 \times 15^2 \times 0.975\right)\right]. \tag{6.18}$$

As previously, we consider a mesh five times finer as one which we could expect to give similar accuracy to our θ=5%, h=0.06 cm case. The same calculation for this finer mesh with $p = 1$ gives

$$(5^3 \times P) \times 2 \times 15^2, \tag{6.19}$$

where the 5^3-term represents the approximate scaling of P under the refinement. The ratio of (6.18) to (6.19) is approximately 5%, which gives us a very large speed up in the cost of applying the preconditioner.

These calculations indicate that it is reasonable to expect an increase in simulation speed of around an order of magnitude for whole-heart simulation.

Chapter 7

Discussion and Evaluation

We have now seen that adaptive high-order FEM improves the efficiency of cardiac simulation in the single-thread case, and that we can expect to see improvements in the case of parallel simulation. We now discuss some of the choices and assumptions that our scheme makes, and recap some of the most important results from the previous chapters.

7.1 The Cell Model

Our treatment of the cell model differs in both concept and in practice from the approach used by many other workers; ours is necessary for a high-order treatment. We now comment on and justify the less common aspects of this.

7.1.1 Assumptions on the Regularity of the Cell Model

Note that the requirement in Theorem 5.2.1 that $\boldsymbol{f}$ be Lipschitz in $\boldsymbol{u}$ is not satisfied for general cell models. However, if we suppose that $\boldsymbol{u}$ is bounded, the Lipschitz property will hold. For simulations, there is evidence to suggest this boundedness occurs for cell models modified to take electroporation currents into account; such currents occur for membrane potentials around 300-400 mV [146, 66, 147, 148]. Indeed, if we do not have

boundedness, we can say that there is a problem in the simulation study design rather than in the numerics; it is certain that real-world quantities such as the transmembrane potential will be bounded in general, and all the more stringently whilst the cell is still functional. We also need to assume that the $[Ca^{2+}]$ is bounded away from zero; again this is likely to be the case in a functioning cell due to leak currents.

7.1.2 Representation of I_{total}

The approach that we take to I_{total} could be compared to a more standard scheme which used a piecewise linear I_{total}. Neither approach involves making any statements about what I_{total} looks like; even though I_{total} is smooth on elements and has discontinuous derivative at element boundaries, the positioning of the elements themselves is essentially arbitrary within the domain. It is clear therefore that no claims are really being made about any lack of smoothness in I_{total}. Therefore, the assertion that we make in using the p-version that the smoothness of u (depending on the smoothness of I_{total}) is sufficient so as not to limit the rate of convergence the p-version is not really any different from the assumptions that are implicitly made when discretising Ω for any standard application of FEM to the cardiac equations.

7.1.3 Gauss Points for the Cell Model

Another approach to approximating the cell model variables w would be to construct a nodal basis using the Gauss points [118]. Instead of integrating the right-hand side of Equation (5.3) using the mass matrix, we could do so by computing the current at each node and then integrating it against each basis function on each element using quadrature. This would work, but would be considerably more computationally demanding than the matrix-based assembly we use [39].

Another alternative would be to use the same nodal basis for u and for w; this would allow matrix-based assembly, but without the need for the change of basis (see Section

5.2.2). However, without the use of the hierarchical basis for u, our method would no longer be suitable for adaptivity.

7.1.4 Choosing the Cell Model Degree, $\tilde{p}$

We discussed this point briefly in Section 5.2.2, and we expand on it here. Technically, the requirements for choosing $\tilde{p}$ are that it is at least as large as the degree being used for u on each element for convergence reasons (see Section 5.2.2), and that

$$\tilde{p} \leq p \tag{7.1}$$

so that our approximation to I_{total} can be integrated using matrix-based assembly. At first glance, these two appear to form two inequalities which require us to choose $\tilde{p} = p$; indeed, this is the choice we make in this work for simplicity. However, there is some subtlety here which is worth expanding upon; note that it would be possible to construct a second mass matrix $\boldsymbol{M}_w$ associated with a solution space S_{hq} larger than our maximal solution space S_{hp} (i.e. $q > p$) and use $\boldsymbol{M}_w$ to integrate our approximation to I_{total}. In this scenario, the restriction given in Inequality (7.1) becomes $\tilde{p} \leq q$. Thus, there is no analytical reason why we cannot make $\tilde{p}$ as large as we wish.

This point has some implications for efficient scheme implementation; because of the lack of spatial derivatives in the PDEs for w, solving for w on each time-step reduces to solving for its value at the nodes of the Lagrange basis, and the solve at each node is independent of the solves at every other node. (This is the reason why the cell models are often thought of as ODE systems scattered throughout Ω, but this way of thinking about them does not make sense in the finite element framework; they must be everywhere-defined.) The lack of spatial coupling means that the nodes of the Lagrange basis must be dense enough in Ω to properly capture the spatial distribution of the behaviour of w. In the cases where we work with large elements, we may have sufficient spatial accuracy in u using S_{hp}, but require $\tilde{p} > p$ in order to have sufficient density of Lagrange basis

nodes. We comment further on this in Section 7.2.2, and it is certainly worth future investigation.

7.2 Evaluation of Uniform Degree Experiments

7.2.1 Summary of Results

We have shown how to successfully employ high-order finite element methods for simulation of the cardiac monodomain system (1.1), meeting or exceeding theoretical error convergence rates. The method achieves our goal of improved efficiency over linear finite elements; we produce highly accurate numerical solutions cheaply, as can be seen for the homogeneous conductivity by comparing the finest linear solution to the second coarsest cubic Δt=0.01 ms solution in Table 5.2. In this case, taking the finest quartic solution as a reference, we see that we obtain a six times smaller percentage error in activation time with a linear system that takes less time to solve (these results are highlighted in the table in blue). It can be used with isotropic and anisotropic conductivity, and geometries which include microstructure such as blood vessels passing through the tissue. The improved efficiency is key because good numerical approximations can take days to obtain with present technology.

As shown in Table 5.2, convergence in the conduction velocity requires very fine meshes when working with p=1. Even at $h = 0.0111$ cm, the CV error is still around 2.5%; at that level, the error in the position of the wavefront will become large during long-time, whole-heart (large domain) simulations. We note that this is using conductivity towards the higher end of the physiological range; with slower physiological (or even pathological) conductivities the error can be very large at this resolution. For example, at 0.01 cm with conductivity 0.2 S m^{-1} in Table 5.1, we have an error of over 10% when using p=1.

In the case of inhomogeneous conductivity (Table 5.3) the second coarsest cubic solution is the point at which the accuracy starts to beat that of the finest linear solution, and it does so by a considerable margin. We note that in this case, it may not be wise

to use the coarsest mesh for simulation; due to the low conductivity perpendicular to the fibre direction (one-fifth of that parallel to the fibres), we see that the activation times at nodes one and two overshoot the converged activation times; we can see this in the maximum percentage error for this simulation. This is due to the effective mesh size being coarser when the conductivity is lower, introducing error into the solution. This explanation is supported by the fact that node three, being connected to the stimulus site by a straight line to which all the fibres along its length are aligned, does not experience any activation time problems. See also Table 5.1, which shows that the same effect occurs in 1D on very coarse meshes when using a low conductivity equal to that for the slow direction here (see $h = 0.1$, p=4). Because error due to the coarse mesh discretisation of u will be compensated for by the accuracy gained as we increase p, we believe that this effect is due to insufficient resolution in the spatial discretisation of w, as this does not change as we refine p. Further investigation is needed to confirm this.

When holes are present in the domain, the simulations are much slower (Table 5.4) due to the increased number of elements required to mesh around the holes (see Figure 5.10). Here we recommend using, for example, p=2 on the third coarsest mesh for better than double the accuracy using a linear system which can be solved in approximately the same amount of time when compared to the finest mesh with p=1. This accuracy can be seen by the position of the wave fronts in Figures 5.10(c) and 5.10(d). An example high-resolution anatomically-derived heart mesh has $h = 0.0125$ cm [95]; this is comparable with the third coarsest mesh here, where the error remains quite considerable with $p = 1$. See Table 7.1 for more detail and other possible choices of mesh and p.

In three dimensions, working with the Schwarz preconditioner, we have seen that adaptivity can produce a three-times more accurate result with a linear system which can be solved in under one-third of the time; this example is comparing $h = 0.04$ cm, $p = 3$ to $h = 0.01$ cm, $p = 1$ on the small cube mesh using the results in Table 5.6. Making the equivalent comparison for the Niederer test domain in Table 5.7 and taking the $h = 0.02$ cm, $p = 4$ solution as a reference, we see that the $h = 0.05$ cm, $p = 3$

Maximum Elt. Diameter (cm)	Simulation Degree	Time Ratio	Example Error Tolerance Level
0.0296	3	1.19	3%
	4	3.17	0.5%
0.0148	1	0.20	12%
	2	1.16	2%
0.0076	1	1 (by def.)	5%

Table 7.1: Simulations on the mesh with holes in the context of the prescribed error tolerances for percentage activation time error at which they would be acceptable. The average single-iteration linear system solve time ratio (time for simulation)/(time for finest linear simulation) is given. Finest mesh p=4 solution taken to be the "true" solution. Error tolerance is on the worst-case percentage activation time error over the three test nodes used. Full data in Table 5.4.

solution has an activation time error at node eight of 5.5%, whereas the $h = 0.01$ cm, $p = 0.01$ has an error of 11.6%; in this case the linear system can be solved five times faster in the high-order than in the linear element case.

7.2.2 Observations regarding Convergence

We note that Figures 5.5(b) and 5.6(b) show some unexpected behaviour, apparent in Figure 5.5(b) as a crossing over of the errors for the p=3 and p=4 solutions, and in Figure 5.6(b) as a smaller-than-expected error reduction when going from p=3 to p=4 on the coarser meshes.

We are not certain of the cause of this. Possible explanations include it being caused by the error introduced by our approximation of $\Pi_{h\tilde{p}}$ by ι, or that the smoothness k of the cell model is limiting, for example where Lemma 5.2.1 is applied. Simulations using the very simple FitzHugh-Nagumo model [149] instead of LR91 with $\tilde{p}$=4 do not display any problems, neither do $\tilde{p}$=6 simulations with LR91 for p=1-4.

Regardless of this, the efficiency remains compelling, and we note that the time-integrated Sobolev norm spatial error (Figure 5.6(a)) does not appear to be so strongly affected.

Similarly, we observe some interesting behaviour in Table 5.6, where we see that the convergence in the activation times is not monotonic as we increase p on the cube mesh

at a resolution of 0.04 cm. We investigated this further by taking the cube mesh and inverting it (reflecting it in the yz-plane and then translating so that it occupies the same space as the original mesh), and discovered that the issue appears to be caused by the bias introduced by the meshing. Table 7.2 re-presents some of the results shown in Table 5.6, together with identical simulations run on the inverted mesh. Note that on the inverted mesh, the convergence at the final node (the eighth, highlighted), the convergence is monotonic in p on the inverted mesh. However, note that the inversion causes a loss of monotonicity in some of the other test nodes (eg. node 3), supporting this meshing bias hypothesis.

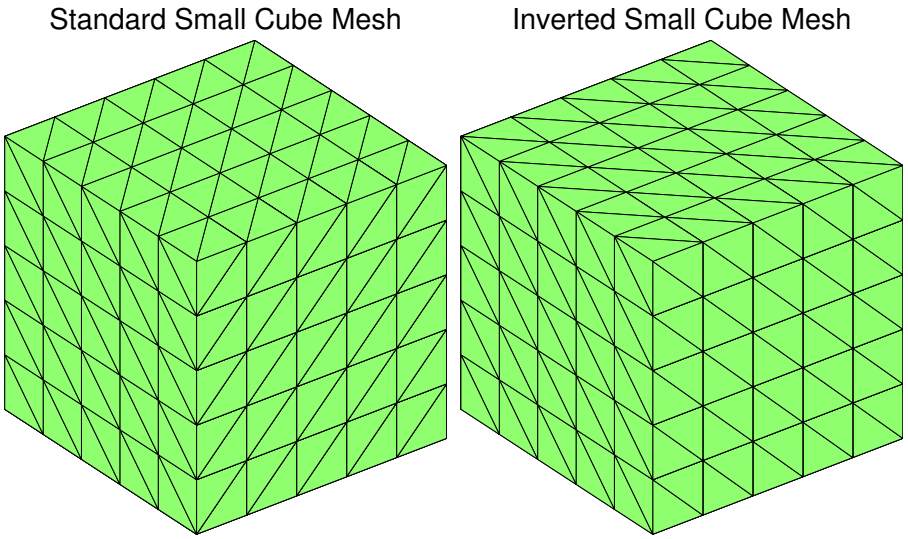


Figure 7.1: A comparison of the element organisation in h=0.04 cm the small cube test mesh used in Sections 5.5.1 and 6.7.1. Inverting the mesh has an effect on the convergence. Note that it will not affect the convergence rate.

Figure 7.1 shows the two meshes, and Figure 5.3 shows the distribution of cell model nodes in an example element. Note that the orientation of the mesh will naturally have an effect on the spatial resolution of the cell model along a line segment joining the centre of the stimulus region to any point at which we measure activation time. As a trivial example, consider the triangle with vertices $(0,0)$, $(0,1)$ and $(1,0)$ cm in the plane; if we have a cell node at each vertex and three more spaced along each edge, the resolution on either of the edges aligned with an axis is 5 cm^{-1}, whereas on the diagonal edge it is ≈ 3.5

cm^{-1}. This drop in resolution can introduce errors, and the effect is more complicated than this in three dimensions.

Mesh (0.04 cm small cube)	Degree p	Activation Time (ms) at Node: 1	2	3	4	5	6	7	8
Original	1	1.29	3.39	6.34	6.34	6.29	7.71	7.94	8.32
	2	1.32	3.77	6.91	6.90	7.29	7.29	8.73	8.93
	3	1.37	3.75	8.08	8.30	8.41	8.77	12.07	12.77
	4	1.33	3.81	8.17	8.03	8.69	8.61	11.54	12.19
Inverted	1	1.35	3.49	6.08	5.72	7.09	6.84	7.02	8.01
	2	1.35	3.70	6.91	6.90	7.57	7.55	8.79	9.31
	3	1.35	3.79	8.10	7.71	8.81	8.43	10.93	11.65
	4	1.35	3.79	7.88	8.10	8.45	8.58	11.64	11.80

Table 7.2: Activation times of the final node in the inverted version of the $0.2 \times 0.2 \times 0.2$ cm small cube domain shown in 5.11 with $h = 0.04$ cm. Fibres and conductivities are as given in [58]. A comparison of the standard and inverted form of this mesh is presented in Figure 7.1. The highlighted column for the eighth node is the one which corresponds to the Final Activation Time column in Table 5.6.

A sensible approach to avoiding this problem would be to increase the cell model degree $\tilde{p}$ further; this would ameliorate the cell resolution problem, and it helped with the unexpected behaviour in one dimension described earlier in this section. Alternatively, the use of an unstructured mesh could spread the effect of the varying cell model resolution more evenly; in this case it is unlikely that inverting the mesh would make any substantial difference to the activation times.

The issues described here, particularly those in three-dimensions, highlight that we must be extremely careful when we decide on a numerical method for solving the monodomain equation. Error can easily be introduced from unexpected sources, and problems would only be identified with thorough investigations such as the one presented in this work. h=0.04 cm meshes such as the one used to generate Table 7.2 have certainly been used for cardiac simulation, and perhaps a clinician would worry about the validity of a simulation in which the order of node activation can be permuted by inverting the mesh (see nodes 5 and 6). In general, one or more of the methods suggested here to avoid this problem should be implemented before working with such a coarse mesh simulation, but doing so would not reduce the efficiency of the method that we have reported. We note

here for completeness that the three-dimensional convergence becomes monotonic in p when the cube mesh is refined to h=0.01 cm elements; the final node activation times in this case, with p increasing from 1 to 4 in order, are 10.11, 11.67, 12.12 and 12.13 ms.

7.3 Evaluation of Adaptivity

Extending our high-order approach, we have demonstrated a method of performing efficient cardiac simulation using spatially-adaptive hierarchical finite elements. The results that we generate provide us with an accuracy that we can have confidence in, as demonstrated by Figure 6.1, where we see that the error indicator does indeed allow the scheme to maintain a target accuracy with remarkable consistency across different meshes. We must emphasise the importance of this plot; it demonstrates just how effective our error control scheme is. Additionally, the target error parameter θ is a useful tool for the user, providing a simple method of tuning the accuracy (and thus time-cost) of a simulation to the needs of a particular study.

7.3.1 Isotropic Simulations in 2D

The most interesting 2D results are in Table 6.1, because in this simulation the wavefront passes through roughly 8 % of the mesh triangles in each millisecond. This percentage is relevant because it is approximately the peak per-millisecond activation percentage for the tetrahedra in a sinus rhythm whole-heart simulation with mean element size 0.025 cm (complete with Purkinje system, which would only boost this number) [150], and so relates to the worst-case wavefront occupancy seen in state-of-the-art scenarios. Our results show that in this case, the adaptive scheme allows us to reduce the error in the simulation to one-fifth of the highest-accuracy linear simulation, and we obtain this result using a linear system which takes a quarter of the time to solve (finest mesh fixed p=1 simulation compared to the 0.1 % tolerance adaptive simulation on the coarsest mesh). Alternatively, we can achieve a comparable error in one-fifth of the time by instead using

0.3 % tolerance on the coarsest mesh. Note that for the realistic simulation, the 8%-per-millisecond peak is very sharp in time, whereas in our test-case we maintain a roughly uniform wavefront occupancy. This means that even greater efficiency improvements should be expected for whole-heart simulation. A graphical presentation of error and DOF ratios is given in Figure 6.3.

Note that the increased efficiency is not surprising, given the narrow region of high-order elements required to effectively capture the steep gradients at the wavefront. This is highlighted in Figure 6.2, which shows the element degree distribution for the case of the coarsest (mean element diameter 0.0442 cm) short narrow mesh with $\theta = 0.1$ %. Comparing the error that this gives to that with linears-only on the same mesh in Table 6.1 shows just how powerful an idea it is to use adaptivity for this problem.

In order to demonstrate more clearly the advantages of the scheme, we also worked with a long, narrow domain $\Omega = [0, 0.25] \times [0, 4]$ cm. The results for this block of simulations are presented in Table 6.2. Here, the wavefront occupancy is lower than in the previous case; this gives us insight into how the scheme would perform away from the peak wavefront occupancy in the realistic case. Here, we see that we can use θ=0.1 % on the coarsest mesh to generate an approximation with one-sixth as much error, but using a linear system that can be solved in one-ninth of the time over the p=1 fixed finest mesh case. For a very high accuracy application, we could use the second coarsest mesh and θ=0.1 % to obtain an error one-thirtieth of the size that the most accurate linear solution produces, doing so with a linear system that takes one-quarter of the time to solve.

We also evaluated isotropic simulation where the wavefront is curved, produced using a stimulus applied in the corner of the square domain. Data on this are presented in Table 6.3. We see similar improvements in efficiency and accuracy in this case: a three-fold improvement in accuracy with a five-times faster linear system, comparing the $\theta = 0.1$ % simulation on the coarsest mesh to the finest fixed linear case.

7.3.2 Anisotropic Simulations in 2D

Anisotropy, and specifically the low conductivity that is associated with it *in vivo*, can cause significant accuracy problems even on very fine meshes. It is therefore interesting to observe the improvements which are possible using adaptivity.

For the results show in Table 6.4, we see that it is possible to increase the accuracy of the solution by an order of magnitude using a linear system which can be solved twice as fast compared to the finest linear scheme ($\theta = 0.1\%$). Graphical representation of the error and DOF ratios is provided by Figure 6.5(a).

When we have holes present in the domain, the results are similar to the fibres-only case. Table 6.5 shows that with $\theta = 0.1\%$, we can achieve a ten-fold improvement in accuracy with a linear system which can be solved in less than half the time, again when compared to the finest linear solution. The error and DOF ratios are shown in Figure 6.5(b).

We expect that once the limitations in the cell model resolution discussed in Section 7.2.2 have been dealt with, we will be able to perform properly converging simulations on the coarser mesh, leading to even more impressive efficiency improvements. This could be done by using an anisotropic mesh structure guided by σ or by increasing $\tilde{p}$, thus increasing the spatial resolution of the cell model, see Section 7.1.4.

7.3.3 Simulations in 3D: The Small Cube and the Niederer Cuboid

The test on the small cube presented in Section 6.7.1 allowed us to perform a simple but thorough investigation of the adaptive scheme in three dimensions. It showed that adaptive high-order elements on a coarse mesh are considerably more computationally efficient than $p = 1$ elements on a very fine mesh. However, the speed advantages for similar accuracy in Table 6.6 compared to using a fixed $p = 3$ or 4 scheme on the coarse mesh are quite modest. While the parameter θ gives us more control over the

error achieved than we would have when limited to a fixed p, as can be seen in Figure 6.9, for suitable p and θ giving similar accuracy, the PCG times are comparable to the non-adaptive method on the same mesh. This is to be expected; as can be seen from Figure 6.8, the wavefront occupancy proportion is very high for such a small mesh, so the difference in the number of degrees of freedom between a fixed and an adaptive scheme is not large. Having said this, both adaptive and non-adaptive high-order approaches outperform the finest linear simulation.

Applying the adaptive scheme to the much larger Niederer test domain allowed us to observe the performance when the wavefront occupancy proportion is far lower. In this case, we see that the PCG times for the adaptive scheme represent a marked improvement on the non-adaptive; we can achieve $p = 4$ accuracy on the $h = 0.05$ cm mesh in around one-third of the time by using $\theta = 1\%$. Comparing the $h = 0.05$ cm mesh to the $h = 0.01$ cm $p = 1$ simulation, the $\theta = 15\%$ case shows an eleven-fold speed-up for similar accuracy at the eighth node, and a seven-fold speed-up better accuracy at all the nodes tested, apart from one which remains comparable, when $\theta = 5\%$. This discrepancy at one node is minor, and is likely because it is connected to the stimulus region by a straight line which has the lowest possible conductivity; the use of anisotropic meshes, as would make sense in general in electrophysiology simulation, would likely solve this problem. There is also likely a mesh orientation issue here, similar to that discussed in Section 7.2.2, which could be mitigated using an unstructured meshing of the domain.

7.3.4 Simulations in 3D: The Rabbit Heart

Due to the non-parallel nature of our software, it has not been possible to perform a thorough evaluation using the rabbit heart mesh. However, in Section 6.8, we presented a simulation on this mesh using $\theta = 5\%$. This serves to prove that the scheme can be applied to realistic geometries, and that it behaves as expected in this case. We do not have a reference solution in this case, so we can not evaluate the error experimentally, but we should expect that we achieve a similar degree of accuracy in this case (mean

$h = 0.06$ cm) to the $h = 0.05$ cm Niederer evaluation.

The estimates of relative computational demand that we make in Section 6.8 provide an evaluation of the maximum increase in speed that we could expect for whole-heart simulation. In terms of FLOPS for the linear system, we could expect a twenty-fold speed up over the state-of-the-art, especially given the highly-parallel nature of the additive Schwarz preconditioner. While it is unlikely that we would achieve the full twenty-fold improvement in practice, realising even half of it would be an important gain.

7.3.5 General Evaluation relative to a High Accuracy Requirement

We wish to say something about the general performance of our scheme, regardless of the domain Ω that we use. One way of doing this is to ask about the relative cost of achieving a target error on each domain, comparing the state-of-the-art linear approach to our adaptive scheme. For example, if we decide that we want a 1% error in the activation times in our numerical approximation, we can make comparisons regarding the mean linear system times required to obtain this; we do so in Table 7.3, which shows benefits regardless of Ω. Note that on two of the meshes it is not possible to obtain an error satisfying the 1% target using linear FEM only. This is interesting for two reasons: firstly because the meshes used are already far finer than those commonly used for cardiac electrical simulation; certainly finer meshes are not used for whole-heart studies. Secondly, it is interesting to look at the magnitude of the error that remains in these linear simulations; it can be around the 4% mark, widely missing the target. See Tables 6.4 and 6.5. This suggests that considerable further refinement may be required to achieve 1% error, and thus the inequalities given in the final column are very loose bounds on the actual value.

We were unable to perform a similar analysis in 3D as the errors remained large even on the finest meshes; none of our 3D simulations was able to achieve a $p = 1$ activation

time error better than 11%.

Domain	Fixed $p = 1$ Time for $< 1\%$ Error (s)	Best Adaptive Time for $< 1\%$ Error (s)	Time Ratio Fixed / Adaptive
Short Narrow ($\Omega = [0, 0.25] \times [0, 1]$)	0.0222	0.0039	5.69
Long Narrow ($\Omega = [0, 0.25] \times [0, 4]$)	0.0917	0.0078	11.75
Square no Fibres ($\Omega = [0, 1] \times [0, 1]$)	0.0712	0.0101	7.04
Square with Fibres ($\Omega = [0, 1] \times [0, 1]$)	Not Achieved (0.0055 cm)	0.0280	> 2.32
Holes with Fibres ($\Omega \subset [0, 1] \times [0, 1]$)	Not Achieved (0.0076 cm)	0.0425	> 2.24

Table 7.3: Comparison of the per-linear-system-solve times taken to achieve a 1% target activation time error by the fixed $p = 1$ linear finite element approach with those for our adaptive scheme. Note that in two cases, even on meshes much finer than those routinely used in cardiac simulation (finest mesh sizes used displayed in brackets), the non-adaptive method fails to achieve this goal; in this case, the ratio is given as an upper bound using the most accurate linear solution time in place of the 1% error time. This failure of the state-of-the-art to obtain this small target error shows that our scheme is not only efficient, but in some cases is the only realistic way of obtaining high-accuracy solutions. In order from the top, the four domains referred to are those with data given in Tables 6.1, 6.2, 6.4 and 6.5, respectively.

7.4 Evaluation of Methods

7.4.1 Parallel Simulation

The investigation that we have been able to perform has been limited by the fact that we do not have a parallel implementation. Because of this, in order to provide a meaningful evaluation, we have only considered the timings for the linear system solve, as this is the component that does not scale straightforwardly in parallel. As noted in Section 6.6, this means that our timings do not include the time taken to evaluate the residual error indicator (Step 3 of the algorithm presented in Section 6.3). Similarly, and as appropriate for adaptive or uniform degree simulations, none of our timings include the time taken to solve for the non-diffusing state variables w of the cell model, nor the time to switch the basis for I_{total} from Lagrangian to hierarchical (see 5.2.2). Because all of these are local problems on an element or sub-element scale, their time-costs will become very small very quickly as we increase the number of processors being used; updates for w can take

less than 5 % of the total simulation time in parallel [139]. In current parallel cardiac simulators, it is the linear system solve time which dominates, due in part to this local nature of the cell model updates, justifying our focus on linear system timings. As future computing hardware will only become more parallel, this is important; for example, the single-instruction, multiple-data nature of GPUs mean that they will be well-suited to dealing with these many identical local problems [151, 67]; see also Section 5.3.2.

Having said this, the locality of the additive Schwarz preconditioner means that the domain can be partitioned amongst processors which require only a minimal amount of interaction with one another during the PCG iterations, so we may see good parallel scaling; we recall that this class of preconditioners has been seen to exhibit superlinear scaling in some cases [124]. It will be interesting to discover how this affects the relative times for the linear system and for the trivially-parallelisable components.

7.4.2 Timings

Timings in general should be treated with some caution, as there are many factors which can affect them. At the most basic level, solving the linear system using an iterative method requires matrix-vector multiplications (matvecs). With the sparse data structures used in this sort of problem, the cost of each matvec is a function of the number of nonzero entries in each row of the system matrix and of the number of rows it has; note that the last two vary with h and p. Each solve will require multiple matvecs to achieve convergence, with the number needed varying with the condition number of the system matrix. Thus, in addition to depending on h and p, the relative time costs of solving different linear systems will vary according to the preconditioner, the hardware type (parallel architecture, CPU cache size, available memory for pre-caching components of the calculation) and the software implementation. Therefore, calculations based on FLOPS, such as those in Section 6.8 may be a more meaningful measure of computational cost.

7.4.3 Preconditioning

There are many possible choices of preconditioner, and we have only investigated a few in this work. A more complete study on optimal preconditioning could be worthwhile, but we believe that those we have investigated work well. Tables 6.1 and 6.2 for 2D and Table 6.6 in 3D present the mean number of iterations that were required to solve the linear system with PCG on each time-step. We note that there is an increase in the number of iterations required over the p=1 case when we are using higher-order elements; this is to be expected due to the increase in the condition number of the system matrix in this case. However, due to the block preconditioner that we are using in 2D (see Section 6.4.1), and the Schwarz preconditioner in 3D (Section 4.6.3), the increase is moderate. It will be informative to investigate the performance of these preconditioners in the parallel setting once such an implementation is available.

Chapter 8

Conclusions

We make some final comments on the work that this thesis documents.

8.1 Overview

This thesis has provided a design template for a locally p-adaptive finite element scheme which achieves its goals of increasing the efficiency of solving the linear system component of the discretised monodomain equation. It is capable of providing much higher accuracy than the state-of-the-art if necessary, can deal with microstructure, and shows particular benefit in the case of anisotropic conductivities. We have tested the method applied to whole-heart simulation, and predicted that we can expect a ten-fold speed-up in parallel for such geometries. The improvements in speed for two-dimensional simulation tell a similar story, demonstrating up to a ten-fold speed-up when the wavefront is in the domain; recall that when the wavefront is absent from the domain the simulation speed will increase greatly under the adaptivity. Because two dimensional simulation is currently the only practical option for tissue-based population simulations, speed improvements in this case are also important.

In three dimensions, we have worked with a very powerful preconditioner which provides excellent control of the condition number across all tested values of p, and remains just as effective in the adaptive case. It is very naturally parallelisable, which means that it can be expected to perform well in a supercomputer-capable implementation of our method.

We have shown that the cell model must be considered to be a PDE when using the FEM rather than a collection of ODEs. The latter masks the spatial error that the cell model introduces into the transmembrane potential solution, and does not make sense under the homogenisation. This is an important point, particularly given that some workers still talk of the essentially-irrelevant "cell-resolution meshes".

8.2 Uniform Degree Monodomain Simulation

Because of the efficiency we have demonstrated, our work leads us to recommend implementation of higher-order finite elements for cardiac monodomain simulation in parallel, with the expectation that this will improve on the computational efficiency of the state-of-the-art. Using high p provides a method of obtaining highly accurate solutions; we have seen that the inaccuracies which occur with $p = 1$ simulation on meshes with commonly-used resolutions can be quite substantial.

We have evaluated the use of high-order finite elements for simulation in one, two and three dimensions, and have demonstrated that they are efficient in all three settings. Of particular interest is the very large gains in accuracy which they provide over linear FEM simulation, even when compared with linear simulations on what are widely considered to be very fine (and thus accuracy-conducive) meshes; this is particularly apparent when an anisotropic conductivity tensor is in use, leading to the delivery of very poor accuracy by linear FEM. In such cases, it is likely that obtaining even larger gains will be possible by using our method together with meshes which are finer perpendicular to the fibre orientation than parallel to it in order to compensate for the reduced CV in that direction,

complemented by using a much higher degree cell model representation. Because the error that can exist in state-of-the-art linear FEM solutions is considerable, we would like to recapitulate that workers must be aware of this problem and should take steps to ensure that their simulation results meet accuracy standards that they consider to be appropriate for their application.

8.3 p-Adaptive Monodomain Simulation

Further to the uniform-degree case, we have demonstrated an adaptive high-order finite element scheme for simulating cardiac electrical propagation. Because of the highly localised steep-gradient behaviour that is characteristic of this problem, this allows us to carefully focus computational effort where it is most needed in space, eliminating computational waste. We have demonstrated efficacy in various two- and three-dimensional test cases.

The scheme can be fine-tuned using the error tolerance parameter θ, allowing us to have confidence in the accuracy of the results we generate; the quality of the error control we use is excellent, as shown by Figure 6.1, and it has the potential to allow researchers to tune the error in their solutions in accordance with the needs of their work. Due to the high-order finite elements that the adaptive scheme uses, we have been able to show that the adaptive scheme has the capability to produce far more accurate solutions than piecewise-linear FEM, and that it is more efficient again than the uniform-degree p-version.

We note that we have only evaluated the situation where the wavefront is present in the domain, and that when it is not present the scheme will generally keep the elements at their lowest degree; for example, certainly between action potentials, and especially for human pacing studies where the cycle length is around 1000 ms but the wavefront is in the domain for only a fraction of that time. The resulting small linear system in the absence of the wavefront will be extremely cheap to solve. We discussed the

rabbit heart analogue of this in Section 6.8, and identified that the whole-heartbeat cycle cost in terms of FLOPS for solving the linear system will be approximately 5% of that for a $p = 1$ simulation with similar accuracy. This opens up the potential for considerable speed improvements in cardiac studies which require the simulation of many heart-beats. Examples include investigations into heterogeneity in slow-phase adaptation in response to changes in pacing cycle length [144] and many-simulation parameter sweeps investigating the effects of a novel drug on cardiac tissue.

8.4 Further Work

8.4.1 Parallelisation and Geometry

We have presented a thorough evaluation of the proposed numerical methods, but due to the details of the implementation have not been able to experimentally evaluate its parallel performance. Developing proof-of-concept software in MATLAB is extremely convenient; designing new algorithms always involves both coding mistakes and conceptual errors which must be dealt with, and MATLAB's workspace interface allows easier examination and repair of such problems. Designing the software which has been used in this work directly in C++ would have been extremely difficult simply due to the increased amount of time that bug-hunting would take. However, the lack of a versatile parallel processing capability in MATLAB is a barrier to a complete evaluation of a real-world applicable (i.e. supercomputer-capable) implementation. Extending the powerful open-source, parallel, C++ cardiac FEM package Chaste [152] to use high-order FEM would be a good route to testing the method in parallel, and this should be a future direction in which this work is taken.

Once this implementation is available, one of the most important tasks will be to analyse the convergence of the method on realistic geometries, building on the whole-heart simulations shown in Section 6.8. In such cases, the use of isoparametric elements should be investigated in order to allow for capturing the detailed domain boundaries at

the epi- and endocardium without having to resort to using very small elements. The computational demands of generating non-adaptive simulation results for comparison purposes will not be possible without this multi-core software. This implementation will allow us to discover how effectively we can precondition this problem in parallel and investigate optimal load-balancing strategies, and it will allow for evaluation of the efficiency gains it provides in the case of practical simulation studies.

8.4.2 The Bidomain

An easy extension of this work which has not been possible for this thesis due to time constraints is to evaluate these methods applied to the bidomain equations. The potential here is great: the bidomain system describes the time-evolution of both an intracellular and extracellular potential field as opposed to the monodomain's single transmembrane potential difference field. The two bidomain fields have differing wavefront steepnesses, and so allowing for different basis degrees to be set for the intracellular and extracellular potentials could allow for even greater gains. This can be further extended to a recently-developed tridomain system [12].

8.4.3 Cell Model Resolution

In this thesis, we have seen indications that the spatial discretisation of the cell model can sometimes be insufficient on very coarse meshes, even when using high-order elements for w. This is something which needs further investigation; it seems likely that there is a sampling problem here. Because the cell models are in practice solved at sufficient discrete points within each element to allow for the construction of a degree-$\tilde{p}$ polynomial, this amounts to sampling of the transmembrane potential. Without a sufficient sampling frequency, we cannot expect there to be enough information captured to contain the original signal. It would be fruitful to investigate this further.

The problem of cell model resolution is closely related to the need for anisotropic

meshes. The elements themselves should have similar anisotropy to the local conductivity tensor; they should be longer in the directions of greater conductivity. Isotropic elements are a computational waste, as they must over-resolve parallel to the fibres when the conductivity is anisotropic.

8.4.4 Time Adaptivity

We would like to see a similar degree of careful study and numerical analysis applied to the development of efficient time-integration schemes for the monodomain system. This could be done using a coupled space-time error indicator, or be based upon the time-adaptive scheme of Perego and Veneziani [85].

8.5 Final Remarks

It is clear that there is a great requirement for enhanced simulation capabilities when it comes to cardiac modelling. The possibility of rapid simulation on smaller-scale computers or clusters will enable researchers in the fields of drug discovery, medical device design, clinical treatment planning, and basic research. The efficiency gains which we have demonstrated in this thesis, once implemented for large-scale simulation in a compiled language, will likely allow us to perform studies with a high degree of accuracy in less time than was previously possible. This will have important implications, not just because it will enable researchers to answer questions about pathological cardiac conditions more quickly, but also because it will help in the design and testing of the cardiac cell models themselves. For example, it has been noted that cell models are rarely validated thoroughly in tissue across a range of physiological and pathological parameters [153], something which would require a large number of simulations; many of these would be for long periods of time. In general, it seems likely that any real insight provided by cardiac electrophysiology simulation studies will come either from such studies based on populations of models, or from simulation that has become so fast that it is possible to

run hundreds of realistic-geometry simulations interactively in an afternoon, encouraging informative manual exploration. In the words of Trefethen [154],

> "A computational study is unlikely to lead to real scientific progress unless the software environment is convenient enough to encourage one to vary parameters, modify the problem, play around."

Perhaps a more succinct statement would be the following. Without efficient simulation capability, cardiac simulation will struggle to reach its full potential.

Appendix A

Basis Functions

We present here the list of the basis functions that we use in this work. Those basis functions presented with a $\pm$ symbol are those which are sometimes used with inverted sign in order to enforce continuity across element boundaries, see Section 4.5.1.

A.1 One Dimensional Basis Functions

The following single-element components of one-dimensional basis functions are defined on a reference element $[0, 1]$. These functions have scaling factors c_i to ensure that their maximum in absolute value is one; this is required with small meshes to avoid singular system matrices due to floating-point error.

1. x (linear uphill)
2. $1 - x$ (linear downhill)
3. $c_2 x(1 - x)$ (quadratic)
4. $c_3 x(1 - x)\left(\frac{1}{2} - x\right)$ (cubic)
5. $c_4 x(1 - x)\left(\frac{1}{3} - x\right)\left(\frac{2}{3} - x\right)$ (quartic)
6. $c_5 x(1 - x)\left(\frac{1}{4} - x\right)\left(\frac{1}{2} - x\right)\left(\frac{3}{4} - x\right)$ (quintic)

7. $c_6 x\,(1-x)\left(\frac{1}{5}-x\right)\left(\frac{2}{5}-x\right)\left(\frac{3}{5}-x\right)\left(\frac{4}{5}-x\right)$ (sextic)

A.2 Two Dimensional Basis Functions

The following single-element components of two-dimensional basis functions are defined on a reference element given by the triangle in the plane with vertices $(0,0)$, $(1,0)$ and $(0,1)$.

Linears:

1. x
2. y
3. $1-x-y$

Quadratics:

1. xy
2. $y(1-x-y)$
3. $x(1-x-y)$

Cubics:

1. $\pm xy\left(x-\frac{1}{2}\right)$
2. $\pm y\left(y-\frac{1}{2}\right)(1-x-y)$
3. $\pm x\left(\frac{1}{2}-x\right)(1-x-y)$
4. $xy(1-x-y)$

Quartics:

1. $xy\left(\frac{1}{3}-x\right)\left(y-\frac{1}{3}\right)$
2. $y(1-x-y)\left(y-\frac{1}{3}\right)\left(y-\frac{2}{3}\right)$

3. $x(1-x-y)\left(x-\frac{1}{3}\right)\left(x-\frac{2}{3}\right)$

4. $x^2y(1-x-y)$

5. $y^2x(1-x-y)$

A.3 Three Dimensional Basis Functions

The following single-element components of three-dimensional basis functions are defined on a reference element given by the tetrahedron in $\mathbb{R}^3$ with vertices $(0,0,0)$, $(1,0,0)$, $(0,1,0)$ and $(0,0,1)$.

Linears:

1. x

2. y

3. z

4. $1-x-y-z$

Quadratics:

1. $x(1-x-y-z)$

2. $y(1-x-y-z)$

3. $z(1-x-y-z)$

4. xy

5. yz

6. xz

Cubics:

1. $yz(1-x-y-z)$

2. $xz(1-x-y-z)$

3. $xy(1-x-y-z)$

4. xyz

5. $\pm x(1-x-y-z)(1-2x-y-z)$

6. $\pm y(1-x-y-z)(1-x-2y-z)$

7. $\pm z(1-x-y-z)(1-x-y-2z)$

8. $\pm xy(x-y)$

9. $\pm yz(y-z)$

10. $\pm xz(z-x)$

Quartics:

Remark. *Only two out of each set of three quartic functions with indices 1-3, 4-6, 7-9 and 10-12 are used on any given element, but all three are required to be available on the reference element. This is because for a general mesh we do not know in advance which two of these will be required to enforce continuity across each particular element boundary. See Section 4.5.1 for more details.*

1. $\pm -zy(1-x-y-z)(1-2z-x-y)$

2. $\pm zy(1-x-y-z)(y-z)$

3. $\pm zy(1-x-y-z)(1-2y-x-z)$

4. $\pm xz(1-x-y-z)(z-x)$

5. $\pm xz(1-x-y-z)(1-2z-x-y)$

6. $\pm -xz(1-x-y-z)(1-2x-y-z)$

7. $\pm - xy(1-x-y-z)(1-2y-x-z)$

8. $\pm xy(1-x-y-z)(x-y)$

9. $\pm xy(1-x-y-z)(1-2x-y-z)$

10. $\pm - xyz(x-y)$

11. $\pm - xyz(y-z)$

12. $\pm - xyz(z-x)$

13. $x(1-x-y-z)(1-y-z-4x)\left(1-y-z-\frac{4}{3}x\right)$

14. $y(1-x-y-z)(1-x-z-4y)\left(1-x-z-\frac{4}{3}y\right)$

15. $z(1-x-y-z)(1-x-y-4z)\left(1-x-y-\frac{4}{3}z\right)$

16. $\frac{1}{3}xy(x-3y)(3x-y)$

17. $\frac{1}{3}yz(z-3y)(3z-y)$

18. $\frac{1}{3}xz(x-3z)(3x-z)$

19. $xyz(1-x-y-z)$

Bibliography

[1] P Goldblatt and R Chappell, editors. *Health Statistics Quarterly*, chapter Death registrations in England and Wales, 2005: causes. Palgrave Macmillan, 2006.

[2] G R Mirams, Y Cui, A Sher, M Fink, J Cooper, B M Heath, N C McMahon, D J Gavaghan, and D Noble. Simulation of multiple ion channel block provides improved early prediction of compounds' clinical torsadogenic risk. *Cardiovasc Res*, 91:53–61, 2011.

[3] W S Redfern, L Carlsson, A S Davis, W G Lynch, I MacKenzie, S Palethorpe, P K S Siegl, I Strang, A T Sullivan, R Wallis, A J Camm, and T G Hammond. Relationships between preclinical cardiac electrophysiology, clinical QT interval prolongation and torsade de pointes for a broad range of drugs: evidence for a provisional safety margin in drug development. *Cardiovasc Res*, 58:32–45, 2003.

[4] I R Efimov, V P Nikolski, and G Salama. Optical imaging of the heart. *Circ Res*, 95:21–33, 2004.

[5] L Tung. *A bi-domain model for describing ischemic myocardial D-C potentials*. PhD thesis, Massachusetts Institute of Technology, USA, 1978.

[6] J Keener and J Sneyd. *Mathematical Physiology*. Springer, 1998.

[7] S A Niederer, L Mitchell, N P Smith, and G Plank. Simulating human cardiac electrophysiology on clinical time-scales. *Front Physio*, 2:Article 14, 2011.

[8] M O Bernabeu, M J Bishop, J Pitt-Francis, D J Gavaghan, V Grau, and B Rodríguez. High Performance Computer Simulations for the Study of Biological Function in 3D Heart Models Incorporating Fibre Orientation and Realistic Geometry and Para-Cellular Resolution. *Comput Cardiol*, 35:721–724, 2008.

[9] Cardiac Chaste: developing software for realistic heart simulations. `http://www.cs.ox.ac.uk/chaste/cardiac_index.html`. [Online; accessed 30-October-2012].

[10] A A Mirin, D F Richards, J N Glosli, E W Draeger, B Chan, J Fattebert, W D Krauss, T Oppelstrup, J J Rice, J A Gunnels, V Gurev, C Kim, J Magerlein, M Reumann, and H Wen. Toward real-time modeling of human heart ventricles at cellular resolution: simulation of drug-induced arrhythmias. In Jeffrey K. Hollingsworth, editor, *SC*, page 2. IEEE/ACM, 2012.

[11] Lawrence Livermore National Laboratory Research Highlights. `https://str.llnl.gov/Sep12/pdfs/9.12.3.pdf`. [Online; accessed 30-October-2012].

[12] A Corrias, P Pathmanathan, D J Gavaghan, and M L Buist. Modelling tissue electrophysiology with multiple cell types: applications of the extended bidomain framework. *Integr Biol*, 4:192–201, 2012.

[13] M O Bernabeu, P Pathmanathan, J Pitt-Francis, and D Kay. Stimulus protocol determines the most computationally efficient preconditioner for the bidomain equations. *IEEE Trans Biomed Eng*, 57:2806–2815, 2010.

[14] M O Bernabeu and D Kay. Scalable parallel preconditioners for an open source cardiac electrophysiology simulation package. *Procedia Comput Sci*, 4:821–830, 2011.

[15] J P Whiteley. Physiology driven adaptivity for the numerical solution of the bidomain equations. *IEEE Ann Bio-Med Eng*, 35:1510–1520, 2007.

[16] E M Cherry, H S Greenside, and C S Henriquez. Efficient simulation of three-dimensional anisotropic cardiac tissue using an adaptive mesh refinement method. *Chaos*, 13:853–865, 2003.

[17] P Colli Franzone, P Deuflhard, B Erdmann, J Lang, and L F Pavarino. Adaptivity in Space and Time for Reaction-Diffusion Systems in Electrocardiology. *SIAM J Sci Comput*, 28:942–962, 2006.

[18] M Pennacchio. The Mortar Finite Element Method for the Cardiac "Bidomain" Model of Extracellular Potential. *J. Sci. Comput.*, 20(2):191–210, 2004.

[19] I Babuška, B A Szabo, and I N Katz. The p-Version of the Finite Element Method. *SIAM J Numer Anal*, 18:515–545, 1981.

[20] I Babuška and M Suri. The p and h-p versions of the finite element method, an overview. *Comput Method Appl M*, 80:5–26, 1990.

[21] I Babuška and M Suri. The p and h-p versions of the finite element method, basic principles and properties. *SIAM Rev*, 36:578–632, 1994.

[22] P. Colli Franzone, L.F. Pavarino, and B. Taccardi. Simulating patterns of excitation, repolarization and action potential duration with cardiac Bidomain and Monodomain models. *Mathematical Biosciences*, 197(1):35–66, 2005.

[23] G Plank, L J Leon, S Kimber, and E J Vigmond. Defibrillation Depends on Conductivity Fluctuations and the Degree of Disorganization in Reentry Patterns. *Journal of Cardiovascular Electrophysiology*, 16(2):205–216, 2005.

[24] N Trayanova. Defibrillation of the heart: insights into mechanisms from modelling studies. *Experimental Physiology*, 91(2):323–337, 2006.

[25] J Pitt-Francis, P Pathmanathan, M O Bernabeu, R Bordas, J Cooper, A G Fletcher, G R Mirams, P Murray, J M Osborne, A Walter, S J Chapman, A Garny, I M M van Leeuwen, P K Maini, B Rodríguez, S L Waters, J P Whiteley, H M Byrne, and D J Gavaghan. Chaste: A test-driven approach to software development for biological modelling. *Computer Physics Communications*, 180(12):2452–2471, 2009. 40

YEARS OF CPC: A celebratory issue focused on quality software for high performance, grid and novel computing architectures.

[26] R Bordas, B Carpentieri, G Fotia, F Maggio, R Nobes, J Pitt-Francis, and J Southern. Simulation of Cardiac Electrophysiology on Next-Generation High-Performance Computers. *Phil Trans R Soc A*, 367:1951–1969, 2009.

[27] F Maggio, B Carpentieri, and G Fotia. A 3D Solver for the Bidomain Equations Based on Unstructured All-hexahedra Spectral Elements. Technical Report 09/50, CRS4, Center for Advanced Studies, Research and Development in Sardinia, Cagliari, Italy, Aug 2009.

[28] F Maggio, B Carpentieri, and G Fotia. Spectral Elements Applied to Cardiac Simulation: A Comparison with Finite Elements. Technical Report 09/51, CRS4, Center for Advanced Studies, Research and Development in Sardinia, Cagliari, Italy, 2009.

[29] F Casadei, E Gabellini, G Fotia, F Maggio, and A Quarteroni. A mortar spectral/finite element method for complex 2D and 3D elastodynamic problems. *Comput Method Appl M*, 191:5119–5148, 2002.

[30] Heart diagram by Wikimedia user ZooFari. `http://en.wikipedia.org/wiki/File:Heart_diagram-en.svg`. [Online; accessed 27-November-2012].

[31] C Guyton and J E Hall. *Textbook of Medical Physiology, 11th Edition*. Saunders, 2006.

[32] R Bordas. *Multiscale Modelling of the Cardiac Specialised Conduction System*. PhD thesis, Univeristy of Oxford, 2011.

[33] R M Berne and M N Levy. *Physiology, International Second Edition*. Mosby, 1988.

[34] C Luo and Y Rudy. A model of the ventricular cardiac action potential. depolarization, repolarization, and their interaction. *Circ Res*, 68:1501–1526, 1991.

[35] M Courtemanche, R J Ramirez, and S Nattel. Ionic mechanisms underlying human atrial action potential properties: insights from a mathematical model. *Am J Physiol Heart Circ Physiol*, 275:301–321, 1998.

[36] R L Winslow, J Rice, S Jafri, E Marbán, and B O'Rourke. Mechanisms of altered excitation-contraction coupling in canine tachycardia-induced heart failure, II: Model studies. *Circ Res*, 84:571–586, 1999.

[37] T R Shannon, F Wand, J Puglisi, C Weber, and D M Bers. A mathematical treatment of integrated Ca dynamics within the ventricular myocyte. *Biophys J*, 87:3351–3371, 2004.

[38] Scholarpedia (Flavio H Fenton and Elizabeth M. Cherry). Scholarpedia, 3(8):1868, 2008. [Online; accessed 5-January-2012].

[39] P Pathmanathan, M O Bernabeu, R Bordas, J Cooper, A Garny, J M Pitt-Francis, J P Whiteley, and D J Gavaghan. A numerical guide to the solution of the bidomain equations of cardiac electrophysiology. *Prog Biophys Mol Bio*, 102:136–155, 2010.

[40] S Linge, J Sundnes, M Hanslien, G T Lines, and A Tveito. Numerical solution of the bidomain equations. *Phil Trans R Soc A*, 367:1931–1950, 2009.

[41] E J Vigmond, R Weber dos Santos, A J Prassl, M Deo, and G Plank. Solvers for the cardiac bidomain equations. *Prog Biophys Mol Bio*, 96:3–18, 2007.

[42] M Potse, B Dubé, J Richer, A Vinet, and R M Gulrajani. A comparison of monodomain and bidomain reaction-diffusion models for action potential propagation in the human heart. *IEEE T Biomed Eng*, 53:2425–2435, 2006.

[43] M Bishop and G Plank. Representing Cardiac Bidomain Bath-Loading Effects by an Augmented Monodomain Approach: Application to Complex Ventricular Models. *IEEE Trans Biomed Eng*, 58:1066–1075, 2011.

[44] J P Wikswo Jr., S-F Lin, and R A Abbas. Virtual Electrodes in Cardiac Tissue: A Common Mechanism for Anodal and Cathodal Stimulation. *Biophys J*, 69:2195–2210, 1995.

[45] A L Hodgkin and A F Huxley. A Quantitative Description of Membrane Current and its Application to Conduction and Excitation in Nerve. *J Physiol*, 117:500–544, 1952.

[46] F Ashcroft. *The Spark of Life: Electricity in the Human Body*. Allen Lane, 2012.

[47] D Noble. Successes and failures in modeling heart cell electrophysiology. *Heart Rhythm*, 8:1798–1803, 2011.

[48] D Noble, A Garny, and P J Noble. How the Hodgkin-Huxley equations inspired the Cardiac Physiome Project. *J Physiol*, 590:2613–2628, 2012.

[49] D Noble. A modification of the hodgkin-huxley equations applicable to purkinje fiber action and pacemaker potential. *J Physiol*, 160:317–352, 1962.

[50] D Noble. A modification of the hodgkin-huxley equations applicable to purkinje fiber action and pacemaker potential. `http://models.cellml.org/workspace/noble_1962/@@rawfile/ae0a0838746d4b969102db6171c9dc17e03dd335/hodgkin_1952.png`, 1962. Figure creator: Hanne Nielsen.

[51] G W Beeler and H Reuter. Reconstruction of the Action Potential of Ventricular Myocardial Fibres. *J Physiol*, 268:177–210, 1977.

[52] S Rush and H Larsen. A practical algorithm for solving dynamic membrane equations. *IEEE Trans Biomed Eng*, 25:389–392, 1978.

[53] G W Beeler and H Reuter. Reconstruction of the Action Potential of Ventricular Myocardial Fibres. `http://models.cellml.org/workspace/beeler_reuter_1977/@@rawfile/fdd29a005ffcf9a72d7ef2479cafb864ea1e887a/beeler_reuter_1977.png`, 1977. Figure creator: Hanne Nielsen.

[54] D Noble and Y Rudy. Models of cardiac ventricular action potentials: iterative interaction between experiment and simulation. *Phil Trans R Soc A*, 359:1127–1142, 2001.

[55] C Luo and Y Rudy. A model of the ventricular cardiac action potential. depolarization, repolarization, and their interaction. `http://models.cellml.org/workspace/luo_rudy_1991/@@rawfile/5564da0874ee2745d6aa7fd1676a9af7bad43341/luo_rudy_i_1991.png`, 1991. Figure creator: Hanne Nielsen.

[56] K H W J ten Tusscher, D Noble, P J Noble, and A V Panfilov. A model for human ventricular tissue. *Am J Physiol Heart Circ Physiol*, 286:1573–1589, 2004.

[57] K H W J ten Tusscher and A V Panfilov. Alternans and spiral breakup in a human ventricular tissue model. *Am J Physiol Heart Circ Physiol*, 291:H1088–H1100, 2006.

[58] S A Niederer, E Kerfoot, A P Benson, M O Bernabeu, O Bernus, C Bradley, E M Cherry, R Clayton, F H Fenton, A Garny, E Heidenreich, S Land, M Maleckar, P Pathmanathan, G Plank, J F Rodríguez, I Roy, F B Sachse, G Seemann, O Skavhaug, and N P Smith. Verification of Cardiac Tissue Electrophysiology Simulators using a N-Version Benchmark. *Phil Trans R Soc A*, 369:4331–4351, 2011.

[59] K H W J ten Tusscher and A V Panfilov. Alternans and spiral breakup in a human ventricular tissue model. `http://models.cellml.org/workspace/tentusscher_panfilov_2006/@@rawfile/134dab554f19e855951376b0d9866dced40c15b6/tentusscher_2006.png`, 2006. Figure creator: Hanne Nielsen.

[60] O Berenfeld and J Jalife. Purkinje-Muscle Reentry as a Mechanism of Polymorphic Ventricular Arrhythmias in a 3-Dimensional Model of the Ventricles. *Circ Res*, 82:1063–1077, 1998.

[61] W T Miller and D B Geselowitz. Simulation studies of the electrocardiogram. I. The normal heart. *Circ Res*, 43:301–315, 1978.

[62] M Bishop, P M Boyle, G Plank, D G Welsh, and E J Vigmond. Modelling the Role of the Coronary Vasculature During External Field Stimulation. *IEEE Trans Biomed Eng*, 57:2335–2345, 2010.

[63] M Bishop, D Gavaghan, N A Trayanova, and B Rodriguez. Photon scattering effects in optical mapping of propagation and arrhythmogenesis in the heart. *J Electrocardiol*, 40:75–80, 2007.

[64] R H Keldermann, M P Nash, H Gelderblom, V Y Wang, and A V Panfilov. Electromechanical wavebreak in a model of the human left ventricle. *Am J Physiol Heart Circ Physiol*, 299:134–143, 2010.

[65] H Tandri, S H Weinberg, K C Chang, R Zhu, N A Trayanova, L Tung, and R D Berger. Reversible cardiac conduction block and defibrillation with high-frequency electric field. *Sci Transl Med*, 3:1–9, 2011.

[66] T Ashihara and N A Trayanova. Asymmetry in Membrane Responses to Electric Shocks: Insights from Bidomain Simulations. *Biophys J*, 87:2271–2282, 2004.

[67] E Bartocci, E M Cherry, J Glimm, R Grosu, S A Smolka, and F H Fenton. Toward Real-time Simulation of Cardiac Dynamics. In *CMSB 2011 9th International Conference on Computational Methods in Systems Biology*, pages 103–112, 2011.

[68] C J Arthurs, M J Bishop, and D Kay. Efficient Simulation of Cardiac Electrical Propagation using High Order Finite Elements. *J Comp Phys*, 231:3946–3962, 2012.

[69] M Ethier and Y Bourgault. Semi-implicit time-discretization schemes for the bidomain model. *SIAM J Numer Anal*, 46:2443–2468, 2008.

[70] J Aguado-Sierra, A Krishnamurthy, C Villongco, J Chuang, E Howard, M J Gonzales, J Omens, D E Krummen, S Narayan, R C P Kerckhoffs, and A D McCulloch. Patient-specific modeling of dyssynchronous heart failure: A case study. *Prog Biophys Mol Bio*, 107:147–155, 2011.

[71] H Talbot, C Duriez, H Courtecuisse, J Relan, M Sermesant, S Cotin, H Delingette, et al. Towards Real-Time Computation of Cardiac Electrophysiology for Training Simulator. *STACOM-MICCAI12*, 2012.

[72] E A Heidenreich, J M Ferrero, M Doblaré, and J F Rodríguez. Adaptive macro finite elements for the numerical solution of monodomain equations in cardiac electrophysiology. *Annals of biomedical engineering*, 38(7):2331–2345, 2010.

[73] N Ayache, D Chapelle, F Clément, Y Coudière, H Delingette, J A Désidéri, M Sermesant, M Sorine, and J M Urquiza. Towards model-based estimation of the cardiac electro-mechanical activity from ECG signals and ultrasound images. *LNCS*, 2230:120–127, 2001.

[74] F H Fenton, E M Cherry, H M Hastings, and S J Evans. Multiple mechanisms of spiral wave breakup in a model of cardiac electrical activity. *Chaos*, 12:852–892, 2002.

[75] R L Courant. Variational methods for the solution of problems of equilibrium and vibrations. *Bull Amer Math Soc*, 49:1–23, 1943.

[76] G Pelosi. The finite-element method, part I: R. L. Courant. *IEEE Antenn Propag M*, 49:180–182, 2007.

[77] C Johnson. *Numerical Solution of Partial Differential Equations by the Finite Element Method*. Dover, 1987.

[78] I Babuška and M Suri. The h-p version of the finite element method with quasiuniform meshes. *RAIRO-Math Model Num*, 21:199–238, 1987.

[79] H I Saleheen and K T Ng. A new three-dimensional finite-difference bidomain formulation for inhomogeneous anisotropic cardiac tissues. *IEEE T Biomed Eng*, 45:15–25, 1998.

[80] M L Trew, B H Smaill, D PBullivant, P J Hunter, and A J Pullan. A generalized finite difference method for modeling cardiac electrical activation on arbitrary, irregular computational meshes. *Math Biosci*, 198:169–189, 2004.

[81] M Trew, I Le Grice, B Smaill, and A Pullan. A finite volume method for modeling discontinuous electrical activation in cardiac tissue. *Ann Biomed Eng*, 33:590–602, 2005.

[82] V Jacquemet and C S Henriquez. Finite volume stiffness matrix for solving anisotropic cardiac propagation in 2-D and 3-D Unstructured Meshes. *IEEE T Biomed Eng*, 52:1490–1492, 2005.

[83] M Marsh, S T Ziaratgahi, and R J Spiteri. The secrets to the success of the Rush-Larsen method and its generalisations. *IEEE Trans Biomed Eng*, to appear, 2012.

[84] J Sundnes, R Artebrant, O Skavhaug, and A Tveito. A Second-Order Algorithm for Solving Dynamic Cell Membrane Equations. *IEEE Trans Biomed Eng*, 56:2546–2548, 2009.

[85] M Perego and A Veneziani. An Efficient Generalization of the Rush-Larsen Method for Solving Electro-Physiology Membrane Equations. *Electron T Numer Ana*, 35:234–256, 2009.

[86] F Maggio, J Southern, G Fotia, and G Cuccura. A 3D Spectral Element Solver for the Bidomain Equations. *VPH 2010*, 2010.

[87] M O Bernabeu, R Bordas, P Pathmanathan, J Pitt-Francis, J Cooper, A Garny, D J Gavaghan, B Rodiguez, J A Southern, and J P Whiteley. Chaste: incorporating a novel multi-scale spatial and temporal algorithm into a large-scale open source library. *Phil Trans R Soc A*, 367:1907–1930, 2009.

[88] J Southern, G J Gorman, M D Piggott, and P E Farrell. Parallel anisotropic mesh adaptivity with dynamic load balancing for cardiac electrophysiology. *J Comp Sci*, 3:8–16, 2012.

[89] B Erdmann, J Lang, and R Roitzsch. KARDOS User's Guide. Technical report, Konrad-Zuse-Zentrum Berlin (ZIB), 2002.

[90] M Pennacchio. The Mortar Finite Element Method for the Cardiac Bidomain Model of Extracellular Potential. *J Sci Comp*, 20:191–210, 2004.

[91] P Deuflhard, B Erdmann, R Roitzsch, and G T Lines. Adaptive finite element simulation of ventricular fibrillation dynamics. *Comput Visual Sci*, 12:201–205, 2009.

[92] J Southern, G J Gorman, M D Piggott, P E Farrell, M O Bernabeu, and J Pitt-Francis. Simulation cardiac electrophysiology using anisotropic mesh adaptivity. *J Comp Sci*, 1:82–88, 2010.

[93] M J Berger and J Oliger. Adaptive mesh refinement for Hyperbolic Partial Differential Equations. *J Comp Phys*, 53:484–512, 1984.

[94] E Cherry, H Greenside, and C Henriquez. A space-time adaptive method for simulating complex cardiac dynamics. *Phys Rev Lett*, 84:1343–1346, 2000.

[95] M J Bishop, G Plank, R A B Burton, J E Schneider, D J Gavaghan, V Grau, and P Kohl. Development of an anatomically detailed MRI-derived rabbit ventricular model and assessment of its impact on simulations of electrophysiological function. *Am J Physiol Heart Circ Physiol*, 298:699–718, 2009.

[96] M. Pennacchio and V. Simoncini. Algebraic multigrid preconditioners for the bidomain reaction–diffusion system. *Applied numerical mathematics*, 59(12):3033–3050, 2009.

[97] M. Pennacchio and V. Simoncini. Non-symmetric Algebraic Multigrid Preconditioners for the Bidomain Reaction–Diffusion system. *Numerical Mathematics and Advanced Applications 2009*, pages 729–736, 2010.

[98] Micol Pennacchio and Valeria Simoncini. Fast Structured AMG Preconditioning for the Bidomain Model in Electrocardiology. *SIAM J. Scientific Computing*, 33(2):721–745, 2011.

[99] R W Dos Santos, G Plank, S Bauer, and E J Vigmond. Preconditioning techniques for the bidomain equations. *Lect Notes Comp Sci*, 40:571–580, 2004.

[100] S C Brenner and L R Scott. *The Mathematical Theory of Finite Element Methods*. Springer, 1994.

[101] L F Pavarino and S Scacchi. Multilevel additive schwarz preconditioners for the bidomain reaction-diffusion system. *Siam J Sci Comput*, 31:420–443, 2008.

[102] S Scacchi. A hybrid multilevel Schwarz method for the bidomain problem. *Comput Methods Appl Mech Engrg*, 197:4051–4061, 2008.

[103] R Sebastian, E Heidenreich, L Dux-Santoy, J Rodriguez, J Ferrero, and J Saiz. Modeling drug effects on personalized 3D models of the heart: a simulation study. *Statistical Atlases and Computational Models of the Heart*, pages 222–231, 2010.

[104] B Rodriguez, K Burrage, D Gavaghan, V Grau, P Kohl, and D Noble. The systems biology approach to drug development: application to toxicity assessment of cardiac drugs. *Clinical Pharmacology & Therapeutics*, 88(1):130–134, 2010.

[105] K J Sampson and C S Henriquez. Electrotonic influences on action potential duration dispersion in small hearts: a simulation study. *American Journal of Physiology-Heart and Circulatory Physiology*, 289(1):H350–H360, 2005.

[106] E Vigmond, F Vadakkumpadan, V Gurev, H Arevalo, M Deo, G Plank, and N Trayanova. Towards predictive modelling of the electrophysiology of the heart. *Experimental physiology*, 94(5):563–577, 2009.

[107] N Trayanova, K Skouibine, and P Moore. Virtual electrode effects in defibrillation. *Progress in biophysics and molecular biology*, 69(2-3):387–403, 1998.

[108] J Wu and D P Zipes. Effects of spatial segmentation in the continuous model of excitation propagation in cardiac muscle. *Journal of cardiovascular electrophysiology*, 10(7):965–972, 2007.

[109] B Rodriguez, B M Tice, J C Eason, F Aguel, J M Ferrero, and N Trayanova. Effect of acute global ischemia on the upper limit of vulnerability: a simulation study. *American Journal of Physiology-Heart and Circulatory Physiology*, 286(6):H2078–H2088, 2004.

[110] S A Niederer, P Lamata, G Plank, P Chinchapatnam, M Ginks, K Rhode, C A Rinaldi, R Razavi, and N P Smith. Analyses of the redistribution of work following cardiac resynchronisation therapy in a patient specific model. *PLoS ONE*, 7:e43504, 2012.

[111] S A Niederer, G Plank, P Chinchapatnam, M Ginks, P Lamata, K S Rhode, C A Rinaldi, R Razavi, and N P Smith. Length-dependent tension in the failing heart and the efficacy of cardiac resynchronization therapy. *Cardiovascular research*, 89(2):336–343, 2011.

[112] R A Adams. *Sobolev Spaces*. Academic Press, 1975.

[113] A Quarteroni and A Valli. *Numerical Approximation of Partial Differential Equations*. Springer-Verlag, 1994.

[114] J D Murray. *Mathematical Biology I: An Introduction, Third Edition*. Springer, 2004.

[115] P G Ciarlet. *The Finite Element Method for Elliptic Problems*. SIAM, classics in applied mathematics edition, 2002.

[116] M Munteanu, L F Pavarino, and S Scacchi. A scalable Newton-Krylov-Schwarz method for the bidomain reaction-diffusion system. *SIAM J Sci Comput*, 31:3861–3883, 2009.

[117] J Q Feng. Application of Galerkin Finite-Element Method with Newton iterations in computing steady-state solutions of unipolar charge currents in corona devices. *J Comp Phys*, 151:969–989, 1999.

[118] A S Ackleh, E J Allen, R B Kearfott, and P Seshaiyer. *Classical and Modern Numerical Analysis. Theory, Methods and Practice*. Chapman and Hall CRC, 2010.

[119] B A Szabo. Some Recent Developments in Finite Element Analysis. *Comput Math Appl*, 5:99–115, 1979.

[120] M S Gockenbach. *Understanding and Implementing the Finite Element Method*. SIAM, 2006.

[121] H Elman, D Silvester, and A Wathen. *Finite elements and fast iterative solvers with applications in incompressible fluid dynamics*. Oxford Science Publications, 2005.

[122] A J Wathen and J R Whiteman. Spectral bounds and preconditioning methods using element-by-element analysis for Galerkin finite element equations. *The mathematics of finite elements and applications, VI (Uxbridge, 1987)*, page 157, 1988.

[123] L F Pavarino. Additive schwarz methods for the p-version finite element method. *Numer Math*, 66:493–515, 1994.

[124] L Giraud, A Haidar, and L T Watson. Parallel scalability study of three dimensional additive Schwarz preconditioners in non-overlapping domain decomposition. Technical Report TR-07-03, Virginia Polytechnic Institute, Virginia, USA, Jan 2007.

[125] M Ainsworth and D Kay. Approximation theory for the hp-version finite element method and application to the non-linear Laplacian. *Appl Numer Math*, 34:329–344, 2000.

[126] M Ainsworth and D Kay. The approximation theory for the p-version finite element method and application to non-linear elliptic PDEs. *Numer Math*, 83:351–388, 1999.

[127] V Thomée. *Galerkin Finite Element Methods for Parabolic Problems, Second Edition*. Springer-Verlag, 2006.

[128] J S Hesthaven. From Electrostatics to Almost Optimal Nodal Sets for Polynomial Interpolation in a Simplex. *SIAM J Numer Anal*, 35:655–676, 1998.

[129] M Hanslien, N Holden, and J Sundnes. A note on discontinuous rate functions for the gate variables in mathematical models of cardiac cells. In *International Conference on Computational Science ICCS 2010*, pages 939–944, 2010.

[130] Bradley J Roth. Artifacts, assumptions and ambiguity: Pitfalls in comparing experimental results to numerical simulations when studying electrical stimulation of the heart. *Chaos*, 12:973–981, 2002.

[131] M Gibb, M Bishop, R Burton, P Kohl, V Grau, G Plank, and B Rodriguez. The role of blood vessels in rabbit proagation dynamics and cardiac arrhythmias. *Lect Notes Comp Sci*, 5528:268–276, 2009.

[132] Matlab Mex API Documentation. http://www.mathworks.co.uk/help/matlab/creating-c-c-and-fortran-programs-to-be-callable-from-matlab-mex-files.html. [Online; accessed 22-October-2012].

[133] Khronos Group OpenCL Homepage. http://www.khronos.org/opencl/. [Online; accessed 22-October-2012].

[134] C Johnson and P Hansbo. Adaptive finite element methods in computational mechanics. *Comput Method Appl M*, 101:143–181, 1992.

[135] P Clément. Approximation by finite element functions using local regularization. *ESAIM: Mathematical Modelling and Numerical Analysis-Modélisation Mathématique et Analyse Numérique*, 1975.

[136] J M Melenk and B I Wohlmuth. On residual-based a posteriori error estimation in hp-FEM. *Adv Comput Math*, 15:311–331, 2001.

[137] M Ainsworth and J Tinsley Oden. A posteriori error analysis in finite element analysis. *Comput Methods Appl Mech Engrg*, 142:1–88, 1997.

[138] J M Melenk. hp-Interpolation of Nonsmooth Functions and an Application to hp-A posteriori Error Estimation. *SIAM journal on numerical analysis*, 43(1):127–155, 2005.

[139] G Plank, R A B Burton, P Hales, M Bishop, T Mansoori, M O Bernabeu, A Garny, A J Prassl, C Bollensdorff, F Mason, F Mahmood, B Rodriguez, V Grau, J E Schneider, D Gavaghan, and P Kohl. Generation of histo-anatomically representative models of the individual heart: tools and applications. *Phil Trans R Soc A*, 367:2257–2292, 2009.

[140] A T Papadopoulos, I S Duff, and A J Wathen. A class of incomplete orthogonal factorization methods using given rotations. II: implementation and results. Technical Report NA02/07, University of Oxford, 2002.

[141] D D Streeter, H M Spotnitz, D P Patel, J Ross, and E H Sonnenblick. Fiber orientation in the canine left ventricle during diastole and systole. *Circ Res*, 24:339–47, 1969.

[142] Qing Lou, Wenwen Li, and I R Efimov. The role of dynamic instability and wavelength in arrhythmia maintenance as revealed by panoramic imaging with blebbistatin vs. 2,3-butanedione monoxime. *Am J Physiol-Heart C*, 302:H262–9, 2012.

[143] M S Link, B J Maron, B A VanderBrink, M Takeuchi, N G Pandian, P J Wang, and M Estes. Impact directly over the cardiac silhouette is necessary to produce ventricular fibrillation in an experimental model of commotio cordis. *J Am Coll Cardiol*, 37:649–654, 2001.

[144] A Bueno-Orovio, B M Hanson, J S Gill, P Taggart, and B Rodriguez. In Vivo Human Left-to-Right Ventricular Differences in Rate Adaptation Transiently Increase Pro-Arrhythmic Risk Following Rate Acceleration. *PLoS One*, In press.

[145] A Bueno-Orovio, B M Hanson, J S Gill, P Taggart, and B Rodriguez. Slow adaptation of ventricular repolarization as a cause of arrhythmia? *Proc 7th Int Workshop Biosignal Interpretation*, pages 283–286, 2013.

[146] D K Cheng, L Tung, and E A Sobie. Nonuniform responses of transmembrane potential during electric field stimulation of single cardiac cells. *Am J Physiol Heart Circ Physiol*, 277:351–362, 1999.

[147] K A DeBruin and W Krassowska. Modeling Electroporation in a Single Cell. I. Effects of Field Strength and Rest Potential. *Biophys J*, 77:1213–1224, 1999.

[148] W Krassowvska. Effects of Electroporation on Transmembrane Potential Induced by Defibrillation Shocks. *PACE*, 18:1644–1660, 1995.

[149] R FitzHugh. Impulses and Physiological States in Theoretical Models of Nerve Membrane. *Biophys J*, 1:445–466, 1962.

[150] R Bordas, K Gillow, Q Lou, I R Efimov, D Gavaghan, P Kohl, V Grau, and B Rodriguez. Rabbit-specific ventricular model of cardiac electrophysiological function including specialized conduction system. *Progress in Biophysics and Molecular Biology*, 107(1):90–100, 2011.

[151] E J Vigmond, P M Boyle, L J Leon, and G Plank. Near-real-time simulations of biolelectric activity in small mammalian hearts using graphical processing units. In *31st Annual International Conference of the IEEE EMBS*, pages 3290–3293, 2009.

[152] Cancer, Heart and Soft Tissue Environment (Chaste). http://www.cs.ox.ac.uk/chaste/. [Online; accessed 3-September-2012].

[153] E M Cherry and F H Fenton. Realistic cardiac electrophysiology modelling: are we just a heartbeat away? *J Physiol*, 588:2689, 2010.

[154] Nick Trefethen's Maxims about Numerical Mathematics, Science, Computers and Life on Earth. http://people.maths.ox.ac.uk/trefethen/maxims.html. [Online; accessed 17-December-2012].