An assessment of numerical methods for cardiac simulation

A Thesis Submitted to the
College of Graduate Studies and Research
in Partial Fulfillment of the Requirements
for the degree of Master of Science
in the Department of Mathematics and Statistics
University of Saskatchewan
Saskatoon

By
Megan E. Marsh

Permission to Use

In presenting this thesis in partial fulfilment of the requirements for a Postgraduate degree from the University of Saskatchewan, I agree that the Libraries of this University may make it freely available for inspection. I further agree that permission for copying of this thesis in any manner, in whole or in part, for scholarly purposes may be granted by the professor or professors who supervised my thesis work or, in their absence, by the Head of the Department or the Dean of the College in which my thesis work was done. It is understood that any copying or publication or use of this thesis or parts thereof for financial gain shall not be allowed without my written permission. It is also understood that due recognition shall be given to me and to the University of Saskatchewan in any scholarly use which may be made of any material in my thesis.

Requests for permission to copy or to make other use of material in this thesis in whole or part should be addressed to:

Head of the Department of Mathematics and Statistics
142 McLean Hall
106 Wiggins Road
University of Saskatchewan
Saskatoon, Saskatchewan
Canada
S7N 5E6

Abstract

Solving the mathematical models of the electrical activity in the heart is difficult mainly due to the complexity of the models required to capture the electrochemical details of the organ. A variety of mathematical models has been developed to describe the electrical activity of individual heart cells. Cardiac cells respond to an electrical stimulus, causing ions to flow across the cell membrane, changing the electrical potential difference between the interior and the exterior of the cell. Cardiac cell models describe the potential difference across the cell membrane, and depending on the complexity of the model, the ion concentrations and the movement of ions through the cell membrane.

This thesis studies 37 cardiac cell models and compares the efficiency of the Forward Euler and Rush–Larsen methods, as well as two generalized Rush–Larsen methods (of order one and order two). The Backward Euler method is compared for six of the 37 models and type-insensitive methods are compared for four of the 37 models. From the results, it is determined that the Rush–Larsen method is the most efficient for moderately stiff models and that the generalized Rush–Larsen methods perform well on the stiff models. The type-insensitive methods are more efficient than the single methods for each of the four models considered.

The bidomain model combines a cardiac cell model with two partial differential equations that model the propagation of the electrical activity throughout the entire heart. Simulations of the bidomain model are computationally expensive, necessitating improvements to the numerical methods used to solve the model. In order to determine whether the cell model results can be applied to the bidomain model, a one-dimensional simulation of the bidomain model is considered and solved using the Chaste software package developed by the Computational Biology Group at Oxford University Computing Laboratory, together with additions written for this thesis. The cellular electrical activity within the bidomain model is modelled by eight of the 37 cell models previously studied, chosen to represent a range of the available models. From these results, it is shown that the cell model results are directly applicable to the one-dimensional bidomain problem. The first-order generalized Rush–Larsen method drastically reduces computation time for the two stiffest models, and the Rush–Larsen method performs optimally for the moderately stiff models. One of the de facto standards, the Forward Euler method, is shown to perform poorly for seven of the eight one-dimensional bidomain simulations.

Acknowledgements

I wish to thank my supervisor, Raymond Spiteri, for encouraging my interest in numerical analysis, for his financial support, for his advice, and for providing me with opportunities to travel and visit other research institutions. Thanks to my partner and husband, Chris Marsh, for listening, offering advice and helping with problems I could not solve. Thanks to my parents, Tom and Chris Lewis, for their emphasis on the value of education and for their emotional and financial support during this undertaking. Thanks to my grandfather, Denis Brown, for encouraging my love of mathematics and for taking such an interest in my research. Thanks also to Philip and Shirley Marsh, for their support and advice when my own parents were unavailable. Thanks to the other members of the Numerical Simulation Lab for their critiques of my work and their friendship. Thanks to NSERC, MITACS, and the Department of Mathematics and Statistics for their financial support and to the Applied Mathematics group for their encouragement. The University of Saskatchewan High Performance Computing group also deserves thanks for their help in getting my software installed on the Gidaspow cluster. Finally, I would like to thank the Chaste Cardiac Team for their advice on my additions to their software and for their help getting the software installed on a variety of machines and operating systems.

Contents

List of Tables

List of Figures

List of Abbreviations

ECG	Electrocardiogram
ODE	Ordinary Differential Equation
PDE	Partial Differential Equation
IVP	Initial–Value Problem
BVP	Boundary–Value Problem
IBVP	Initial–Boundary–Value Problem
RHS	Right Hand Side
LHS	Left Hand Side
FE	Forward Euler
BE	Backward Euler
RK	Runge–Kutta
RL	Rush–Larsen
GRL	Generalized Rush–Larsen
GRL1	Generalized Rush–Larsen method of order one
GRL2	Generalized Rush–Larsen method of order two
A-stable	Absolutely stable
RRMS	Relative Root Mean Square
MRMS	Mixed Root Mean Square
TI	Type–insensitive
ms	millisecond
HH	Hodgkin–Huxley
FHN	FitzHugh–Nagumo
LR	Luo–Rudy
FDM	Finite Difference Method
FEM	Finite Element Method

Chapter 1

Introduction

The heart is one of the most important organs in the human body. The pumping of blood through the heart transports essential nutrients and chemicals throughout the body and removes waste products from the cells. Heart disease affects these vital functions by damaging the heart and is one of the leading causes of death in Canada, with 22% of deaths in 2007 attributed to heart disease [9]. Due to the prevalence of heart disease, furthering our understanding of this essential organ is necessary for improving the diagnosis of heart conditions. Because abnormalities in the electrical activity in the heart are linked with heart disease, there is particular interest in expanding the understanding of the electrical activity in the heart [61].

The effect of the electrical activity on the heart's health makes its study a natural choice for aiding in diagnosing heart conditions. However, due to the heart's major role in maintaining life, it is difficult to study a living heart. An increasingly practical approach is to develop mathematical models that describe the electrical processes in the heart and to use computer simulations to solve the models. The electrical activity of cardiac cells can be modelled by a system of ordinary differential equations. The equations describe the evolution of the potential difference across the cell membrane and depending on the complexity of the cell model, describe the evolution of the ion concentrations within a cell and the flow of the ions through the cell membrane. The propagation of the electrical activity in the heart can be modelled using the bidomain model [66]. The bidomain model uses a system of partial differential equations coupled with a cardiac cell model to describe the propagation of the electrical activity in the heart.

An interactive model of a patient's heart can be used by a clinician as a guide during an invasive procedure. For example, such a guide would be useful when a pacemaker is installed because the simulation could give information as to the optimal location to install the device. Such an interactive system would require real-time simulation [35]. However, the computational time required to solve a realistic simulation prevents real-time simulation. At the time of writing, the most efficient whole heart simulation remains approximately 240 times slower than real time even using 16,384 computer cores [35]. To achieve real-time whole heart simulations, it is necessary to further improve the numerical methods used to solve the model.

This thesis focuses on improving the numerical methods used to solve the cardiac cell mod-

els, solved separately and within the bidomain model, in order to reduce the computational time required for cardiac simulations. Cardiac cell simulations are solved within Matlab [32], a commercially available problem-solving environment equipped with a high-level programming language. Bidomain simulations are solved with the Cancer, Heart, and Soft Tissue Environment (Chaste) [45], a C++ software package for large-scale heart simulation.

1.0.1 Structure of thesis

The remainder of this thesis is structured into four chapters. Chapter 2 provides background information on the physiology of the heart, an introduction to mathematical models of the electrical activity of the heart, an introduction to the operator splitting method for solving differential equations, and an overview of some of the software packages available for cardiac simulation. Chapter 3 provides an overview of necessary terminology, introduces a new error norm, and describes the numerical methods used to solve ordinary differential equations and partial differential equations. It also describes numerical methods used by current software packages, including Chaste, to solve cardiac cell models and the bidomain model and provides an overview of the contributions made by this thesis to the Chaste software package. Chapter 4 demonstrates the effectiveness of the newly introduced error norm, describes the simulations, and provides results. Chapter 5 describes conclusions and future work. Appendix A provides additional cardiac cell model results.

Chapter 2

Background

2.1 Physiology

2.1.1 Physiology of the heart

The heart is a muscular pump made up of four pumping chambers: the right and left atria, located near the base of the heart, and the right and left ventricles, located near the apex of the heart. The chambers are connected by atrioventricular valves that, when open, allow blood to move from the atria to the ventricles. A fibrous skeleton separates the four pumping chambers and acts as an electrical insulator, separating electrically active myocytes, or muscle cells, located throughout the heart [26].

The pumping of the heart is activated by an electrical stimulus. A band of specialized myocardial cells located near the base of the heart, called the sinoatrial (SA) node, spontaneously depolarize, causing a wave of depolarization [26]. The wave of depolarization is propagated to the atrial myocardium, causing the atria to contract, increasing the atrial pressure and the blood flow from the atria to the ventricles. Following the atrial contraction, the pressure drops and the atrioventricular valves are pulled upwards by the decreased pressure. The wave of depolarization is propagated through the atrioventricular (AV) bundle from the atrial myocardium to the ventricles. The propagation of the depolarization wave through the ventricles is synchronized by the Purkinje system, allowing the ventricles to contract in a coordinated manner. The contraction of the ventricles increases the intraventricular pressure, forcing blood into the aorta and the pulmonary arteries. Figure 2.1 is a simplified diagram that shows the four pumping chambers, the SA node and the AV node, as well as the route of the electrical conduction through the heart.

2.1.2 Cardiac cells

The myocardium, composed of myocytes and connective tissue, makes up the majority of the heart's thickness [26]. There are several types of cardiac myocytes, including working myocytes in the atria and ventricles, specialized for contraction, Purkinje fibers, specialized for rapid conduction, and nodal cells in the SA and AV nodes, specialized for pacemaker activity and atrioventricular delay

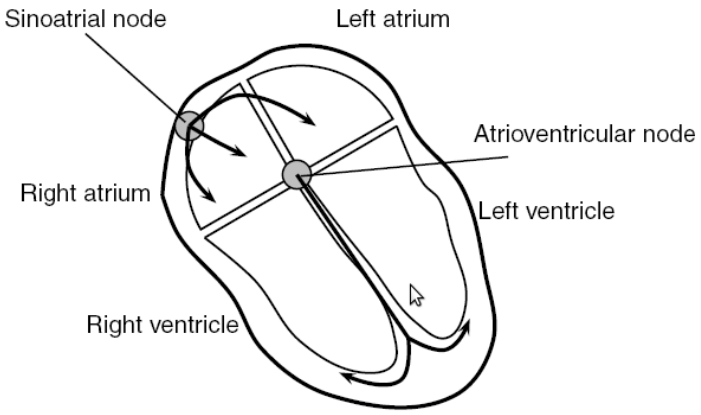

Figure 2.1: Simplified diagram of the heart and the pumping chambers, taken from Chapter 1 of [61]. The electrical activation begins in the sinoatrial node and propagates through the atria, the atrioventricular node, and the ventricles.

[26]. Cardiac myocytes contain charged particles (ions) that cause a potential difference across the cell membrane, called the transmembrane potential. The ions allow the cells to respond to an electrical stimulus, changing their transmembrane potential [61]. Without an electrical stimulus, the transmembrane potential of a cell remains at a resting potential. However, once the transmembrane potential is raised above a threshold value (dependent on the type of cardiac cell), the conductive properties of the cell change, allowing for ions to flow across the cell membrane. The potential difference across the membrane changes rapidly with the flow of ions, causing a depolarization of the membrane. Once the membrane is depolarized, the cell membrane gradually repolarizes, returning the transmembrane potential to the initial resting potential value [61]. Because of gap junctions, channels between cardiac myocytes that allow ions to move freely between adjacent cells, it is possible for electrical impulses to be transmitted rapidly throughout the heart [26]. The action potential of a cardiac cell is this process of depolarization followed by a repolarization stage returning the transmembrane potential to the initial resting potential.

2.2 Models

2.2.1 The inverse problem in cardiology

The typical goal of a physician is to examine a patient and to accurately diagnose the cause of the exhibited symptoms. In the case of cardiology, this generally translates into studying the electrocardiogram (ECG), i.e., the graph of the electrical potential on the surface of the patient's body. Ideally, a cardiologist inputs the data from the ECG into a model that determines the cause of the symptoms based on the input data. Finding the electrical activity in the heart by measuring

the electrical potential on the surface of the body is the inverse problem in electrocardiology.

When numerically solving mathematical problems, it is assumed that the problem is well-posed. A well-posed problem has the following properties [61]:

1. A solution exists.

2. The solution is unique.

3. The solution depends continuously on the data.

When a problem fails to satisfy one of the above conditions, it is called an ill-posed problem. Inverse problems are typically ill-posed, including the inverse problem in electrocardiology [61]. Fortunately, the forward problem in electrocardiology, namely simulating the electrical activity in the heart, is well-posed [61]. Being able to solve the forward problem aids in finding solutions to the inverse problem. Chapter 7 of [61] describes methods for solving the inverse problem using solutions to the forward problem, including that of modelling myocardial infarctions (heart attacks) using data from the ECG. This thesis focuses on numerically solving the forward problem.

2.2.2 Cell models

We begin by modelling the electrical activity of a single cardiac cell. A cardiac cell model consists of ordinary differential equations (ODEs) that, at a minimum, describe the evolution of the electrical potential difference across the cell membrane and typically also describe the movement of ions across the cell membrane and the concentration of ions within the cell. One simple cell model, the FitzHugh–Nagumo (FHN) cell model [17], consists of two ODEs that describe only the potential difference across the cell membrane and a recovery variable. By modelling only two variables, the solution to the FHN model does not accurately reflect the action potential of a heart cell; rather it gives only a general idea of the action potential. Increasingly accurate physiologically based cell models, such as the Luo–Rudy I cell model [30] or the model of Winslow et al. [69], are able to capture greater detail, such as the calcium concentration and the flow of ions across the cell membrane.

Luo Rudy I model

As an example of a cardiac cell model, consider the Luo–Rudy I (LR) cell model. The LR model uses eight variables to model the action potential of a guinea pig ventricular cell. The variables considered are the transmembrane potential V_m, the intracellular calcium concentration Ca_i, and six non-dimensional gating variables, m, h, j, d, f, and X, that control the movement of ions across

the cell membrane. The ordinary differential equations are given by

$$\begin{aligned}\frac{dV_m}{dt} &= -\frac{1}{C}(I_{Na} + I_{si} + I_K + I_{K1} + I_{Kp} + I_b),\\ \frac{dCa_i}{dt} &= -10^{-4} I_{si} + 0.07(10^{-4} - Ca_i),\\ \frac{dy}{dt} &= \frac{(y_\infty - y)}{\tau_\infty},\end{aligned}$$

where y is one of the gating variables m, h, j, d, f, or X, $y_\infty = y_\infty(V_m)$, and $\tau_\infty = \tau_\infty(V_m)$. The currents I_{Na}, I_{si}, I_K, I_{K1}, I_{Kp}, and I_b are nonlinear functions of V_m, Ca_i, m, h, j, d, f, and X. For complete details of the LR model, see [30].

2.2.3 Bidomain model

The bidomain model, first proposed by Tung in 1978 [66], is a continuum-based model used to relate the electrochemical activity of individual heart cells with the electrical activity across the entire heart. The bidomain model considers the heart to consist of two superimposed domains. The first domain, the intracellular domain, consists of the components inside the cell membrane of the heart cells. The second domain, the extracellular domain, consists of the components outside the cell membrane of the heart cells. The bidomain model is coupled with a cell model describing the reactions and flow of ions across the cell membrane of a heart cell. Because the bidomain model is a continuum-based model, individual heart cells are not modelled, but instead cell models are used to model averages of heart cells. Assuming that the heart is isolated from the surrounding tissue, the bidomain model can be written as [61]

$$\frac{\partial \mathbf{s}}{\partial t} = \mathbf{f}(\mathbf{s}, V_m, t), \tag{2.1a}$$

$$\chi C_m \frac{\partial V_m}{\partial t} + \chi \mathbf{I}_{ion}(\mathbf{s}, V_m, t) = \nabla \cdot (\mathbf{M}_I \nabla V_m) + \nabla \cdot (\mathbf{M}_I \nabla u_E)\,, \tag{2.1b}$$

$$0 = \nabla \cdot (\mathbf{M}_I \nabla V_m) + \nabla \cdot ((\mathbf{M}_I + \mathbf{M}_E)\nabla u_E)\,, \tag{2.1c}$$

with boundary conditions

$$\begin{aligned}\hat{\mathbf{n}} \cdot (\mathbf{M}_I \nabla V_m + \mathbf{M}_I \nabla u_E) &= 0,\\ \hat{\mathbf{n}} \cdot (\mathbf{M}_E \nabla u_E) &= 0,\end{aligned} \tag{2.2}$$

where V_m is the transmembrane potential variable, u_E is the extracellular potential variable, $\mathbf{s}$ is a vector of state variables from the cell model, $\mathbf{M}_I$ and $\mathbf{M}_E$ are conductivity tensors that model the dependency of the conductivity of the heart on the surrounding tissue, χ is the area of cell membrane per unit volume, and C_m is the capacitance of the cell membrane per unit area. The functions $\mathbf{f}$ and $\mathbf{I}_{ion}$ are dependent on the particular cell model used. It is possible to simplify

the bidomain model into the so-called monodomain model by assuming that anisotropy rates are equal, i.e., $\mathbf{M}_I = \lambda \mathbf{M}_E$, where λ is a scalar. This simplifies the equations, making the model easier to analyze and to solve numerically. However, the equal anisotropy rate assumption is not found to hold from measurements of real intracellular and extracellular conductivities and important electrophysiological phenomena are lost when this assumption is used [61].

2.3 Operator splitting

Operator splitting is a technique typically used for dividing a difficult set of differential equations into two or more subproblems that are potentially easier to solve. Operator splitting has been used for the simulation of air pollution [27], the electrical activity in the heart [63], and reaction-diffusion equations [13]. Operator splitting divides a difficult problem into two (or more) subproblems that are solved numerically over the same timestep. By splitting the problem, it is possible to use methods that are optimized for the different components of the problem. It has been shown in [53] that the highest order achievable for an operator splitting method with positive timesteps is of order two. Operator splitting methods of third order and higher are possible, but negative timesteps are required [71]. It is shown in [55] that for linear parabolic PDEs, negative timesteps are not necessarily unstable for all splitting methods. In general, however, negative timesteps may lead to instability for splitting methods [53]; therefore in this thesis we focus on first-order and second-order operator splitting methods. A second-order operator splitting method was first described in [59].

2.3.1 Overview

Following [53], the technique of operator splitting can be described by considering an initial-value problem (IVP) of the form given by

$$\frac{d\mathbf{u}}{dt} = \mathbf{A}\mathbf{u} + \mathbf{B}\mathbf{u}, \qquad 0 < t < T, \tag{2.3a}$$

$$\mathbf{u(0)} = \mathbf{u}_0, \tag{2.3b}$$

where $\mathbf{A}$ and $\mathbf{B}$ are constant matrices that in general do not commute and arise from the spatial discretization of a PDE. The exact solution to (2.3) is $\mathbf{u}(t) = e^{(\mathbf{A}+\mathbf{B})t}\mathbf{u}_0$. We are interested in approximating $e^{(\mathbf{A}+\mathbf{B})t}$ using only products of $e^{\mathbf{A}t}$ and $e^{\mathbf{B}t}$ because $e^{\mathbf{A}t}$ and $e^{\mathbf{B}t}$ are more desirable to treat. Two first-order operator splitting methods used to split $\mathbf{M} = \mathbf{A} + \mathbf{B}$ are given by

$$S(\mathbf{A}, \mathbf{B}) = e^{\mathbf{A}t} e^{\mathbf{B}t}, \tag{2.4}$$

$$S(\mathbf{A}, \mathbf{B}) = e^{\mathbf{B}t} e^{\mathbf{A}t}. \tag{2.5}$$

Two second-order operator splitting methods are given by

$$S(\mathbf{A}, \mathbf{B}) = e^{\mathbf{A}\frac{1}{2}t} e^{\mathbf{B}t} e^{\mathbf{A}\frac{1}{2}t}, \tag{2.6}$$

$$S(\mathbf{A},\mathbf{B}) = e^{\mathbf{B}\frac{1}{2}t}e^{\mathbf{A}t}e^{\mathbf{B}\frac{1}{2}t}. \tag{2.7}$$

The splittings of $\mathbf{M}$ described in (2.4)–(2.7) can be described in terms of the following algorithm, where for definiteness we use (2.4) to solve (2.3) over one timestep Δt:

1. For $0 \leq t \leq \Delta t$, solve:
$$\frac{d\mathbf{v}}{dt} = \mathbf{A}\mathbf{v}, \qquad \mathbf{v}(0) = \mathbf{u}_0, \tag{2.8}$$
to obtain $\mathbf{v}_{\Delta t} = \mathbf{v}(\Delta t)$.

2. For $0 \leq t \leq \Delta t$, solve:
$$\frac{d\mathbf{w}}{dt} = \mathbf{B}\mathbf{w}, \qquad \mathbf{w}(0) = \mathbf{v}_{\Delta t}, \tag{2.9}$$
to obtain $\mathbf{w}_{\Delta t} = \mathbf{w}(\Delta t)$.

Then $\mathbf{w}_{\Delta t}$ is a first-order approximation to the exact solution $\mathbf{u}$ to (2.3) at $t = \Delta t$. The proof of the consistency and order of this method is given in the following section.

2.3.2 Consistency and order

We show that the operator splitting method as described in (2.4) is both consistent and of first order. The proofs for the other methods are similar.

The consistency and order of the operator splitting method can be shown using Taylor series expansions. This is done following [61]. First, approximating the solution to (2.3) $\mathbf{u}$ at time Δt as a Taylor series, we obtain

$$\mathbf{u}(\Delta t) = \mathbf{u}_0 + \Delta t \frac{d\mathbf{u}}{dt}\bigg|_{t=0} + \frac{(\Delta t)^2}{2}\frac{d^2\mathbf{u}}{dt^2}\bigg|_{t=0} + O\left(\Delta t^3\right).$$

Because $\mathbf{A}$ and $\mathbf{B}$ are independent of t, we compute $\frac{d^2\mathbf{u}}{dt^2}$ as

$$\frac{d^2\mathbf{u}}{dt^2} = \frac{d}{dt}\left((\mathbf{A}+\mathbf{B})\mathbf{u}\right) = (\mathbf{A}+\mathbf{B})\frac{d\mathbf{u}}{dt}. \tag{2.10}$$

Substituting $d\mathbf{u}/dt$ in (2.10),

$$\begin{aligned}\frac{d^2\mathbf{u}}{dt^2} &= (\mathbf{A}+\mathbf{B})(\mathbf{A}+\mathbf{B})\mathbf{u},\\ &= (\mathbf{A}+\mathbf{B})^2\mathbf{u},\\ &= (\mathbf{A}^2+\mathbf{A}\mathbf{B}+\mathbf{B}\mathbf{A}+\mathbf{B}^2)\mathbf{u}.\end{aligned}$$

Inserting this term into the Taylor series expansion gives

$$\mathbf{u}(\Delta t) = \mathbf{u}_0 + \Delta t(\mathbf{A}+\mathbf{B})\mathbf{u}_0 + \frac{(\Delta t)^2}{2}(\mathbf{A}+\mathbf{B})^2\mathbf{u}_0 + O(\Delta t^3).$$

Expanding $\mathbf{v}(\Delta t)$ and $\mathbf{w}(\Delta t)$ from (2.8) and (2.9), respectively, gives

$$\begin{aligned}\mathbf{v}(\Delta t) &= \mathbf{u}_0 + \Delta t\mathbf{A}\mathbf{u}_0 + \frac{(\Delta t)^2}{2}\mathbf{A}^2\mathbf{u}_0 + O(\Delta t^3),\\ \mathbf{w}(\Delta t) &= \mathbf{v}(\Delta t) + \Delta t\mathbf{B}\mathbf{v}(\Delta t) + \frac{(\Delta t)^2}{2}\mathbf{B}^2\mathbf{v}(\Delta t) + O(\Delta t^3).\end{aligned}$$

For $\mathbf{w}(\Delta t)$, we get

$$\begin{aligned}\mathbf{w}(\Delta t) &= \left(\mathbf{u}_0 + \Delta t\mathbf{A}\mathbf{u}_0 + \frac{(\Delta t)^2}{2}\mathbf{A}^2\mathbf{u}_0 + O\left((\Delta t)^3\right)\right) \\ &\quad + \Delta t\mathbf{B}\left(\mathbf{u}_0 + \Delta t\mathbf{A}\mathbf{u}_0 + \frac{(\Delta t)^2}{2}\mathbf{A}^2\mathbf{u}_0 + O\left((\Delta t)^3\right)\right) \\ &\quad + \frac{(\Delta t)^2}{2}\mathbf{B}^2\left(\mathbf{u}_0 + \Delta t\mathbf{A}\mathbf{u}_0 + \frac{(\Delta t)^2}{2}\mathbf{A}^2\mathbf{u}_0 + O\left((\Delta t)^3\right)\right) + O\left((\Delta t)^3\right), \\ &= \mathbf{u}_0 + \Delta t(\mathbf{A}+\mathbf{B})\mathbf{u}_0 + \frac{(\Delta t)^2}{2}\left(\mathbf{A}^2 + 2\mathbf{A}\mathbf{B} + \mathbf{B}^2\right)\mathbf{u}_0 + O\left((\Delta t)^3\right).\end{aligned}$$

The error at $t = \Delta t$ is the difference between $\mathbf{w}(\Delta t)$ and $\mathbf{u}(\Delta t)$. This is

$$\begin{aligned}&\mathbf{w}(\Delta t) - \mathbf{u}(\Delta t) \\ &\quad = \left(\mathbf{u}_0 + \Delta t(\mathbf{A}+\mathbf{B})\mathbf{u}_0 + \frac{(\Delta t)^2}{2}\left(\mathbf{A}^2 + 2\mathbf{A}\mathbf{B} + \mathbf{B}^2\right)\mathbf{u}_0 + O((\Delta t)^3)\right) \\ &\qquad - \left(\mathbf{u}_0 + \Delta t(\mathbf{A}+\mathbf{B})\mathbf{u}_0 + \frac{(\Delta t)^2}{2}(\mathbf{A}+\mathbf{B})^2\mathbf{u}_0 + O((\Delta t)^3)\right), \\ &\quad = \frac{(\Delta t)^2}{2}\left(\mathbf{A}\mathbf{B} - \mathbf{B}\mathbf{A}\right)\mathbf{u}_0 + O((\Delta t)^3), \\ &\quad = O((\Delta t)^2),\end{aligned}$$

giving an error proportional to $(\Delta t)^2$ after one step. The number of timesteps n is proportional to $(\Delta t)^{-1}$, making the error at the final time proportional to Δt; hence we have a first-order method. Note that if $\mathbf{A}$ and $\mathbf{B}$ commute, we have a second-order method.

2.3.3 Operator splitting for cardiac modelling

Operator splitting for cardiac modelling is used to simplify the numerical solution of the bidomain model by splitting the system of ODEs, (2.1a), from the two PDEs, (2.1b) and (2.1c). In this way, it is possible to use methods better suited for each respective component. Splitting the system of ODEs from the PDEs can be achieved using any of the schemes (2.4)–(2.7). Assuming a timestep of Δt, initial conditions given by $V_m(\mathbf{x}, t_n) = V_{m,n}$, $u_E(\mathbf{x}, t_n) = u_{E,n}$, and $\mathbf{s}(\mathbf{x}, t_n) = \mathbf{s}_n$, and by applying the splitting algorithm (2.4), the splitting can be done by the following [61]

1. Solve the system of ODEs arising from the cell models for $t \in [t_n, t_n + \Delta t]$, and denote the solutions V_m and $\mathbf{s}$ following this step by $V^1_{m,n+1}$ and $\mathbf{s}^1_{n+1}$:

$$\begin{aligned}\frac{\partial \mathbf{s}}{\partial t} &= \mathbf{f}(\mathbf{s}, V_m, t), \\ \frac{\partial V_m}{\partial t} &= -\frac{1}{C_m}\mathbf{I}_{ion}(\mathbf{s}, V_m, t).\end{aligned}$$

2. Solve the PDEs, using $V^1_{m,n+1}$ and $\mathbf{s}^1_{n+1}$ as the initial condition, for $t \in [t_n, t_n + \Delta t]$, to obtain the final solution $V_{m,n+1}$ and $\mathbf{s}_{n+1}$:

$$\begin{aligned}\chi C_m\frac{\partial V_m}{\partial t} &= \nabla\cdot\left(\mathbf{M}_I\nabla V_m\right) + \nabla\cdot\left(\mathbf{M}_I\nabla u_E\right), \\ 0 &= \nabla\cdot\left(\mathbf{M}_I\nabla V_m\right) + \nabla\cdot\left(\left(\mathbf{M}_I + \mathbf{M}_E\right)\nabla u_E\right),\end{aligned}$$

with boundary conditions

$$\hat{\mathbf{n}} \cdot (\mathbf{M}_I \nabla V_m + \mathbf{M}_I \nabla u_E) = 0,$$
$$\hat{\mathbf{n}} \cdot (\mathbf{M}_E \nabla u_E) = 0.$$

This scheme is known as the first-order Godunov splitting method [19]. Note that by using the scheme (2.5), it is possible to reverse the order in which the systems are solved. We are also interested in the second-order Strang splitting method [59]. By applying the splitting algorithm (2.6), Strang splitting can be done by the following

1. Solve the system of ODEs arising from the cell models for $t \in \left[t_n, t_n + \frac{\Delta t}{2}\right]$, and denote the solutions V_m and $\mathbf{s}$ following this step by $V_{m,n}^{1/2}$ and $\mathbf{s}_n^{1/2}$:

$$\frac{\partial \mathbf{s}}{\partial t} = \mathbf{f}(\mathbf{s}, V_m, t),$$
$$\frac{\partial V_m}{\partial t} = -\frac{1}{C_m}\mathbf{I}_{ion}(\mathbf{s}, V_m, t).$$

2. Solve the PDEs, using $V_{m,n}^{1/2}$ and $\mathbf{s}_n^{1/2}$ as the initial condition, for $t \in [t_n, t_n + \Delta t]$, and denote the solutions V_m and $\mathbf{s}$ following this step by $V_{m,n+1}^{1/2}$ and $\mathbf{s}_{n+1}^{1/2}$:

$$\chi C_m \frac{\partial V_m}{\partial t} = \nabla \cdot (\mathbf{M}_I \nabla V_m) + \nabla \cdot (\mathbf{M}_I \nabla u_E),$$
$$0 = \nabla \cdot (\mathbf{M}_I \nabla V_m) + \nabla \cdot ((\mathbf{M}_I + \mathbf{M}_E) \nabla u_E),$$

   with boundary conditions

$$\hat{\mathbf{n}} \cdot (\mathbf{M}_I \nabla V_m + \mathbf{M}_I \nabla u_E) = 0,$$
$$\hat{\mathbf{n}} \cdot (\mathbf{M}_E \nabla u_E) = 0.$$

3. Solve the system of ODEs arising from the cell models, using $V_{m,n+1}^{1/2}$ and $\mathbf{s}_{n+1}^{1/2}$ as the initial condition, for $t \in \left[t_n + \frac{\Delta t}{2}, t_{n+1}\right]$, to obtain the final solution $V_{m,n+1}$ and $\mathbf{s}_{n+1}$:

$$\frac{\partial \mathbf{s}}{\partial t} = \mathbf{f}(\mathbf{s}, V_m, t),$$
$$\frac{\partial V_m}{\partial t} = -\frac{1}{C_m}\mathbf{I}_{ion}(\mathbf{s}, V_m, t).$$

Again, it is possible to reverse the order in which the systems are solved by using the splitting algorithm (2.7). In practice, Steps 1 and 3 are combined to reduce computational effort, leapfrogging the solution of the system of ODEs and the PDEs until the final time is reached.

2.4 Software

There are several software packages available that solve the monodomain and bidomain models, including the Cardiac Arrhythmia Research Package (CARP) [67], the Montreal heart model

(PROPAG) [46], Simula Research Laboratory's Python Computing Components (PyCC) framework [54], Continuity 6 [20], and the Cancer, Heart, and Soft Tissue Environment (Chaste) [45]. Software by Ying et al. solves the monodomain model [70].

We are using Chaste to solve instances of the bidomain problem. Chaste is a project developed by a team mainly based in the Computational Biology Group at Oxford University Computing Laboratory. The development strategy is to produce heart simulation software that is generic, efficient, accurately tested and validated, and capable of massive simulations in parallel. Their software is open-source C++, making it possible to modify and add methods.

The CellML project [24] is an online repository for storing and sharing validated mathematical models, with emphasis on mathematical models of biological processes. CellML is a markup language that is used to encode the necessary information for a particular mathematical model and a wide variety of cardiac cell models are available from this repository. For more details about the structure of CellML, its current applications, and its future development, see [29]. Chaste has an addition called PyCML, written in Python, that automatically generates the C++ files necessary for modelling a particular cell model in Chaste from the model information available from the CellML repository. This automatic generation of C++ code prevents unnecessary human errors and permits a large number of cardiac cell models to be easily used in Chaste.

Chapter 3

Numerical Methods

Because we cannot expect to solve all instances of the bidomain model analytically, it is necessary to consider numerical solutions to the bidomain model. Operator splitting can be applied to the coupled system of ODEs and PDEs, reducing it to a system of nonlinear ODEs and a system of two linear PDEs, as is described in detail in Chapter 3 of [61]. The method employed by the Chaste software, discussed in Section 3.4.1, also splits the coupled system of ODEs and PDEs into a system of nonlinear ODEs and a system of two linear PDEs. This chapter discusses general methods for solving systems of ODEs and systems of PDEs, with emphasis on the methods used in this thesis to solve the bidomain model. However, before discussing general numerical methods for PDEs and ODEs, it is necessary to define the concepts used in this thesis. The general ODE system

$$\frac{d\mathbf{y}}{dt} = \mathbf{f}(t, \mathbf{y}(t)), \qquad t > 0, \tag{3.1}$$

is considered throughout this chapter. To numerically solve an ODE system, either boundary conditions or initial conditions are required. We are interested in the initial-value problem with an initial condition at $t = 0$,

$$\frac{d\mathbf{y}}{dt} = \mathbf{f}(t, \mathbf{y}(t)), \qquad 0 < t < t_f, \qquad \mathbf{y}(0) = \mathbf{y}_0. \tag{3.2}$$

3.1 Concepts and definitions

The information in this section has largely been summarized from [3], [61], and [52].

3.1.1 Existence and uniqueness

When solving an IVP, we require that a solution exist and be unique. From [3], we have

Theorem 1

Let D be an open connected set in $\mathbb{R}^2$, let $\mathbf{f}(t, \mathbf{y})$ be a continuous function of t and $\mathbf{y}$ for all $(t, \mathbf{y})$ in D, and let $(t_0, \mathbf{y}_0)$ be an interior point of D. Assume that $\mathbf{f}(t, \mathbf{y})$ satisfies a Lipschitz condition

$$\|\mathbf{f}(t, \mathbf{y}_1) - \mathbf{f}(t, \mathbf{y}_2)\| \leq K\|\mathbf{y}_1 - \mathbf{y}_2\| \qquad \forall\, (t, \mathbf{y}_1), (t, \mathbf{y}_2) \in D,$$

for some $K \geq 0$. Then there is a unique function $\mathbf{y}(t)$ defined on an interval $[t_0 - \alpha, t_0 + \alpha]$ for some $\alpha > 0$, satisfying

$$\frac{d\mathbf{y}}{dt} = \mathbf{f}(t, \mathbf{y}(t)), \qquad t_0 - \alpha \leq t \leq t_0 + \alpha,$$
$$\mathbf{y}(t_0) = \mathbf{y}_0.$$

It is assumed throughout this thesis that $\mathbf{f}$ satisfies a Lipschitz condition in order to guarantee that a solution exists and is unique for the IVP.

3.1.2 Local and global error

Two forms of error are typically considered for numerical methods solving IVPs: the local error and the global error. Consider the IVP (3.2) with exact solution $\mathbf{y}(t)$. Suppose a solver takes n steps from $t = 0$ to $t = t_n$ to compute an approximation at t_n, $\mathbf{y}_n \approx \mathbf{y}(t_n)$, and one more step from $t = t_n$ to $t = t_{n+1}$ to compute an approximation at t_{n+1}, $\mathbf{y}_{n+1} \approx \mathbf{y}(t_{n+1})$. Consider also the IVP starting at t_n,

$$\frac{d\mathbf{u}}{dt} = \mathbf{f}(t, \mathbf{u}(t)), \qquad \mathbf{u}(t_n) = \mathbf{y}_n,$$

with exact solution $\mathbf{u}(t)$. The local error at t_{n+1} is given by

$$\mathbf{e}_{\text{local}} = \mathbf{u}(t_{n+1}) - \mathbf{y}_{n+1}.$$

The global error at t_{n+1} is given by

$$\mathbf{e}_{\text{global}} = \mathbf{y}(t_{n+1}) - \mathbf{y}_{n+1}.$$

Typically, solvers directly control the local error by controlling the stepsize, but they only indirectly control the global error through control of the local error.

3.1.3 Error norms

In order to compare the accuracy and efficiency of numerical methods for solving a particular IVP over the interval $t \in [t_0, t_f]$, it is necessary to have a measure of the accuracy of the numerical method. This can be done by computing an average of the error at N points in $t \in [t_0, t_f]$ and comparing these average error values. However, in order to compute an average of the error, either the exact solution must be known or a reference solution must be computed for all N points. A reference solution is a solution to the IVP that is known to have converged to d digits of accuracy at all N points, where d is sufficiently large and determined by comparing increasingly accurate solutions and counting the number of matching digits for all N points. In practice, the magnitude of d is limited by floating point precision and by the time required to compute the reference solution.

We are interested in two error norms. The first norm is called the Mixed Root Mean Square (MRMS) error and is defined by

$$e_{\text{MRMS}} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(\frac{\hat{y}_i - y_i}{1+|\hat{y}_i|}\right)^2}, \tag{3.3}$$

where y_i is the numerical solution and $\hat{y}_i$ is the reference solution at time t_i. The second norm is called the Relative Root Mean Square (RRMS) error and is defined by

$$e_{\text{RRMS}} = \sqrt{\frac{1}{N}\frac{\sum_{i=1}^{N}(\hat{y}_i - y_i)^2}{\sum_{i=1}^{N}\hat{y}_i^2}}, \tag{3.4}$$

where y_i is the numerical solution and $\hat{y}_i$ is the reference solution at time t_i.

The RRMS error norm has previously been used to compare numerical solutions of the trans-membrane potential variable from cardiac cell models, e.g., see [60]. The MRMS error norm is a 2-norm combination of a relative error with an absolute error and has not, to our knowledge, been used previously in heart simulation. A comparison of the two norms for the McAllister et al. (1975) cell model is done in Section 4.2.

3.1.4 Order of convergence

The order of convergence of a given numerical method determines the theoretical rate at which a numerical solution tends towards the true solution as the stepsize approaches zero. This can be described mathematically, following [3]. Consider the IVP

$$\frac{d\mathbf{y}}{dt} = \mathbf{f}(t, \mathbf{y}(t)), \tag{3.5a}$$

$$\mathbf{y}(t_n) = \mathbf{y}_n, \tag{3.5b}$$

where $\Delta t_n = t_{n+1} - t_n$, and suppose $\mathbf{y}(t)$ is the exact solution to (3.5) and suppose $\mathbf{y}_{\Delta t_n}(t)$ is the solution obtained by one step of some numerical method over the interval $t_n < t < t_{n+1}$. Suppose we have, for some constant integer $p \geq 0$,

$$\|\mathbf{y}(t) - \mathbf{y}_{\Delta t_n}(t)\| \leq c(\Delta t_n)^{p+1}, \qquad t_n < t < t_{n+1}, \tag{3.6}$$

where c is some constant and p is the largest integer such that (3.6) holds. Then we say that the numerical method used to obtain $\mathbf{y}_{\Delta t_n}(t)$ is convergent with order p.

3.1.5 Stability

When discussing the stability of IVPs, we must consider both the IVP and the numerical method used to solve the IVP. An IVP is stable if small perturbations in the initial conditions cause small perturbations in the solution. Following [52], suppose we have the IVP (3.2), assume $\mathbf{y}_1(t)$ and

$\mathbf{y}_2(t)$ are solutions to (3.2), and $\mathbf{f}$ satisfies a Lipschitz condition with constant K. For $t_1 < t_2$, we have

$$\|\mathbf{y}_2(t_2) - \mathbf{y}_1(t_2)\| \leq \|\mathbf{y}_2(t_1) - \mathbf{y}_1(t_1)\| e^{K(t_2 - t_1)}. \tag{3.7}$$

Assuming K is of moderate size, we have that (3.2) is moderately stable. It is assumed throughout this thesis that any IVPs discussed are moderately stable. We next consider the stability of the numerical method used to solve the IVP. Consider the general ODE system (3.1). Following [52], we can linearize (3.1) about a point $(t^*, \mathbf{y}^*)$ to obtain the linear, constant-coefficient equation

$$\frac{d\mathbf{u}}{dt} = \mathbf{f}(t^*, \mathbf{y}^*) + \frac{\partial \mathbf{f}}{\partial \mathbf{y}}(t^*, \mathbf{y}^*)(\mathbf{u} - \mathbf{y}^*), \qquad t > 0. \tag{3.8}$$

Because we are interested in the stability of (3.1), i.e., how the solution differs when initial conditions change, we must consider the difference between two solutions of (3.8). Let $\mathbf{v}$ also satisfy (3.1) to yield after linearization

$$\frac{d\mathbf{v}}{dt} = \mathbf{f}(t^*, \mathbf{y}^*) + \frac{\partial \mathbf{f}}{\partial \mathbf{y}}(t^*, \mathbf{y}^*)(\mathbf{v} - \mathbf{y}^*), \qquad t > 0. \tag{3.9}$$

Subtracting equation (3.9) from equation (3.8) and letting $\mathbf{w} = \mathbf{u} - \mathbf{v}$ be the difference between the solutions, we get, for $t > 0$,

$$\begin{aligned}
\frac{d\mathbf{u}}{dt} - \frac{d\mathbf{v}}{dt} &= \left[\mathbf{f}(t^*, \mathbf{y}^*) + \frac{\partial \mathbf{f}}{\partial \mathbf{y}}(t^*, \mathbf{y}^*)(\mathbf{u} - \mathbf{y}^*)\right] - \left[\mathbf{f}(t^*, \mathbf{y}^*) + \frac{\partial \mathbf{f}}{\partial \mathbf{y}}(t^*, \mathbf{y}^*)(\mathbf{v} - \mathbf{y}^*)\right], \\
\frac{d\mathbf{u}}{dt} - \frac{d\mathbf{v}}{dt} &= \frac{\partial \mathbf{f}}{\partial \mathbf{y}}(t^*, \mathbf{y}^*)(\mathbf{u} - \mathbf{v}), \\
\frac{d\mathbf{w}}{dt} &= \frac{\partial \mathbf{f}}{\partial \mathbf{y}}(t^*, \mathbf{y}^*)(\mathbf{w}).
\end{aligned}$$

Assuming $\frac{\partial \mathbf{f}}{\partial \mathbf{y}}(t^*, \mathbf{y}^*)$ is diagonalizable, i.e., there exists a non-singular matrix $\mathbf{T}$ of eigenvectors such that $\mathbf{T}^{-1}\frac{\partial \mathbf{f}}{\partial \mathbf{y}}(t^*, \mathbf{y}^*)\mathbf{T} = \mathbf{\Lambda}$, where $\mathbf{\Lambda}$ is a diagonal matrix of eigenvalues, we write

$$\frac{d\mathbf{w}}{dt} = \mathbf{T}\mathbf{\Lambda}\mathbf{T}^{-1}\mathbf{w}, \qquad t > 0.$$

Letting $\tilde{\mathbf{w}} = \mathbf{T}^{-1}\mathbf{w}$, we get

$$\frac{d\tilde{\mathbf{w}}}{dt} = \mathbf{\Lambda}\tilde{\mathbf{w}}, \qquad t > 0. \tag{3.10}$$

In equation (3.10), we have a fully decoupled system of equations, where each component equation has the form

$$\frac{d\tilde{w}_i}{dt} = \lambda_i \tilde{w}_i, \qquad t > 0.$$

This leads to the *test equation* [52], an equation used to test the stability of a numerical method. The test equation is written as

$$\frac{dy}{dt} = \lambda y, \qquad t > 0, \tag{3.11}$$

$$y(0) = 1. \tag{3.12}$$

We know that the exact solution to (3.11) is

$$y = e^{\lambda t}.$$

For $Re(\lambda) < 0$, the exact solution to the test equation satisfies

$$y(t) \to 0 \text{ as } t \to \infty; \tag{3.13}$$

hence we want a numerical solution applied to (3.11) to satisfy

$$y_{\Delta t}(t_n) \to 0 \text{ as } t_n \to \infty, \tag{3.14}$$

for any stepsize Δt. For a given numerical method, we are interested in the set of values $\lambda \Delta t$ that satisfy (3.14). This set of values located in the complex plane is called the region of absolute stability of a numerical method. A method that has a region of absolute stability that contains the entire left half of the complex plane is called an A-stable method [8]. Examples of the stability regions of several numerical methods are presented later in the chapter.

3.1.6 Explicit and implicit methods

A numerical method used to solve IVPs is categorized as either explicit or implicit. A method for which y_{n+1} is given directly from known quantities and previous values of y_n is called an explicit method. A method for which y_{n+1} is found by solving a system of (nonlinear) equations is called an implicit method because y_{n+1} is defined implicitly. The classic Forward Euler method and the Backward Euler method are explicit and implicit methods, respectively, described in Section 3.2.1. Explicit methods are useful when stepsizes are not restricted by stability conditions because they have a small computational cost per step compared with implicit methods. However, implicit methods typically have larger stability regions, making them useful for solving stiff problems. Stiffness is described in Section 3.1.8.

3.1.7 Jacobian

Given a function $\mathbf{F}(\mathbf{x})$, with $\mathbf{x} = \{x_1, x_2, \ldots, x_n\}^T$ and $\mathbf{F} = \{F_1, F_2, \ldots, F_n\}^T$, the Jacobian matrix of $\mathbf{F}(\mathbf{x})$ is defined by

$$\mathbf{J_F}(\mathbf{x}) = \begin{pmatrix} \frac{\partial F_1}{\partial x_1} & \frac{\partial F_1}{\partial x_2} & \cdots & \frac{\partial F_1}{\partial x_n} \\ \frac{\partial F_2}{\partial x_1} & \frac{\partial F_2}{\partial x_2} & \cdots & \frac{\partial F_2}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial F_n}{\partial x_1} & \frac{\partial F_n}{\partial x_2} & \cdots & \frac{\partial F_n}{\partial x_n} \end{pmatrix}.$$

The Jacobian matrix of the right-hand side (RHS) of a system of ODEs can be used for analysis of the system and is required when a system of ODEs is linearized. The eigenvalues of the Jacobian matrix are also of interest because they can aid in determining the stiffness of a problem, discussed in Section 3.1.8.

3.1.8 Stiffness

Stiffness is a property of an IVP that manifests itself when the IVP is solved numerically. A main feature of a stiff IVP is the restriction on stepsize due to stability and not on accuracy requirements. A stiff IVP typically has an eigenvalue λ with a large negative real part occurring in the Jacobian matrix that forces the timestep Δt of a numerical method to be small so that $|\lambda \Delta t|$ is within the stability region of the numerical method. Implicit methods tend to be used to solve stiff IVPs due to their large stability regions. A large stability region is more conducive to allowing the stepsize to be chosen based on accuracy and not on stability.

3.2 Numerical solution of ODEs

3.2.1 The Forward and Backward Euler methods

The most intuitive method for solving IVPs is arguably the Forward Euler (FE) method. Given the IVP (3.5), for $t_n < t < t_{n+1}$, where $\Delta t = t_{n+1} - t_n$, the FE method approximates (3.5) by

$$\mathbf{y}_{n+1} = \mathbf{y}_n + \Delta t\, \mathbf{f}(t_n, \mathbf{y}_n). \tag{3.15}$$

The FE method (3.15) can be derived as in [61] by integrating (3.5a) from t_n to t_{n+1} and using a left end-point approximation for the integral on the RHS of the equation to obtain

$$\begin{aligned} \int_{t_n}^{t_{n+1}} \frac{d\mathbf{y}}{dt}\, dt &= \int_{t_n}^{t_{n+1}} \mathbf{f}(t, \mathbf{y}(t))\, dt, \\ \mathbf{y}(t_{n+1}) - \mathbf{y}(t_n) &= \int_{t_n}^{t_{n+1}} \mathbf{f}(t, \mathbf{y}(t))\, dt, \\ \mathbf{y}(t_{n+1}) &\approx \mathbf{y}(t_n) + \Delta t\, \mathbf{f}(t_n, \mathbf{y}(t_n)). \end{aligned}$$

The test equation (3.11) is used to determine the stability region of a numerical method. The stability region of the FE method, from [52], is the set

$$S = \{|1 + z| \leq 1, Re(z) \leq 0\}.$$

The FE method is a first-order explicit method and is trivial to implement. However, the FE method has limitations, particularly when problems are stiff.

Another method that is also first order and can be derived in a similar fashion is the Backward Euler (BE) method, as is done in [61]. Given the IVP (3.5), for $t_n < t < t_{n+1}$, where $\Delta t = t_{n+1} - t_n$, the BE method approximates (3.5) by

$$\mathbf{y}_{n+1} = \mathbf{y}_n + \Delta t\, \mathbf{f}(t_{n+1}, \mathbf{y}_{n+1}). \tag{3.16}$$

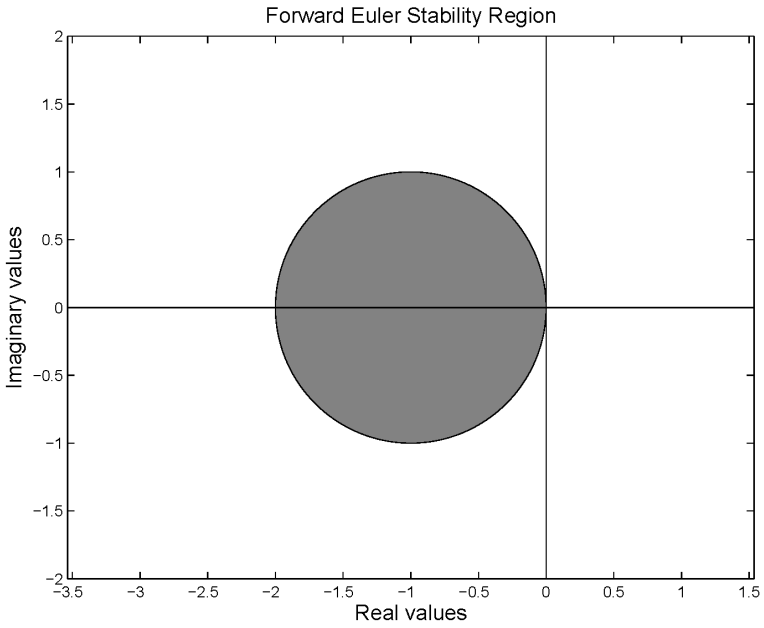

Figure 3.1: Stability region for the FE method.

The BE method (3.16) can be derived by integrating (3.5a) and using a right end-point approximation to the integral on the RHS of the equation to obtain

$$\int_{t_n}^{t_{n+1}} \frac{d\mathbf{y}}{dt}\, dt = \int_{t_n}^{t_{n+1}} \mathbf{f}(t, \mathbf{y}(t))\, dt,$$

$$\mathbf{y}(t_{n+1}) - \mathbf{y}(t_n) = \int_{t_n}^{t_{n+1}} \mathbf{f}(t, \mathbf{y}(t))\, dt,$$

$$\mathbf{y}(t_{n+1}) \approx \mathbf{y}(t_n) + \Delta t\, \mathbf{f}(t_{n+1}, \mathbf{y}(t_{n+1})).$$

The stability region of the BE method, from [52], is the set

$$S = \left\{ \left| \frac{1}{1-z} \right| \leq 1, Re(z) \leq 0 \right\}.$$

The stability region of the BE method contains the entire left half of the complex plane and hence is more stable than the FE method. However, because the BE method is implicit, it generally requires that a system of nonlinear equations from equation (3.16) be solved each step, increasing the typical computational time for a given timestep. Following [56] and assuming a constant stepsize, the algorithm for the BE method is given below.

Backward Euler Algorithm

Input: The RHS of the ODE, $\mathbf{f}(t, \mathbf{y})$, the Jacobian $\mathbf{J_f}(t, \mathbf{y})$ of $\mathbf{f}(t, \mathbf{y})$, the initial time t_0, the final time t_f, the initial condition $\mathbf{y}_0$, the timestep Δt, and the error tolerance `TOL`

Output: An approximate solution $\mathbf{y}$ at the final time t_f

$t = t_0$;

$\mathbf{y}_1 = \mathbf{y}_0$;

Calculate the total number of timesteps: $n_{\text{steps}} = \lceil \frac{t_f - t_0}{\Delta t} \rceil$;

for $n = 1$ to n_{steps} **do**

Use the FE method to compute the initial iterate:

$\mathbf{y}_{n+1}^{(0)} = \mathbf{y}_n + \Delta t\, \mathbf{f}(t_n, \mathbf{y}_n)$;

Set $\nu = 1$;

Set $e = \texttt{TOL} + 1.0$;

while $e > \texttt{TOL}$ **and** $\nu < \nu_{\max}$ **do**

Compute the Newton–Raphson update:

$\mathbf{y}_{n+1}^{(\nu)} = \mathbf{y}_{n+1}^{(\nu-1)} - \mathbf{J}^{-1}(t_n, \mathbf{y}_{n+1}^{(\nu-1)})\mathbf{f}(t_n, \mathbf{y}_{n+1}^{(\nu-1)})$;

Compute the error between iterates and save the maximum value to determine whether the Newton–Raphson method has converged:

$e = \|\mathbf{y}_{n+1}^{(\nu)} - \mathbf{y}_{n+1}^{(\nu-1)}\|_\infty$;

Update the number of iterations that has been completed:

$\nu = \nu + 1$;

end while

if $\nu = \nu_{\max}$ **then**

The Newton–Raphson method failed to converge, exit with failure;

end if

Save the computed solution at time t_{n+1}:

$\mathbf{y}_{n+1} = \mathbf{y}_{n+1}^{(\nu)}$;

$t = t + \Delta t$;

end for

where $\lceil x \rceil = \min\{m \in \mathbb{Z} | m \geq x\}$. In practice, $\nu = 20$, $\texttt{TOL} = 10^{-5}$, the solution $\mathbf{y}_{n+1}$ is saved at each time t_{n+1}, and $\mathbf{J}_{\mathbf{f}}(t, \mathbf{y})$ is approximated using Matlab's `numjac` function.

3.2.2 Runge–Kutta methods

The family of Runge–Kutta (RK) methods generalizes the Euler methods by using increasingly accurate methods to approximate the integral on the RHS of (3.5). This is done by computing intermediate values for $\mathbf{f}(t, \mathbf{y}(t))$ for $t_n \leq t \leq t_{n+1}$ and using a weighted sum of the intermediate values to approximate $\mathbf{y}(t_{n+1})$. A general s-stage RK method computes s intermediate values, and the RHS of (3.5a) is approximated by [61]

$$\int_{t_n}^{t_{n+1}} \mathbf{f}(t, \mathbf{y}(t))\, dt \approx \Delta t \sum_{i=1}^{s} b_i \mathbf{K}_i,$$

where the intermediate stage values, $\mathbf{K}_i$, are given by

$$\mathbf{K}_i = \mathbf{f}\left(t_n + c_i \Delta t, \mathbf{y}(t_n) + \Delta t \sum_{j=1}^{s} a_{ij} \mathbf{K}_j\right), \qquad i = 1, 2, \ldots, s.$$

The values of c_i, a_{ij}, and b_i determine the method and its properties such as the order of accuracy. An RK method can be represented by a Butcher tableau, a construct that lists the coefficients in an efficient manner, such that it is possible to reconstruct the RK method. A general Butcher tableau is given by

$$\begin{array}{c|cccc} c_1 & a_{11} & a_{12} & \cdots & a_{1s} \\ c_2 & a_{21} & a_{22} & \cdots & a_{2s} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_s & a_{s1} & a_{s2} & \cdots & a_{ss} \\ \hline & b_1 & b_2 & \cdots & b_s \end{array}$$

In matrix notation, the Butcher tableau is given by

$$\begin{array}{c|c} \mathbf{c} & \mathbf{A} \\ \hline & \mathbf{b^T} \end{array}$$

As an example, consider Heun's method, a second-order RK method with two stages and given by

$$\begin{aligned} \mathbf{K}_1 &= \mathbf{f}(t_n, \mathbf{y}_n), \\ \mathbf{K}_2 &= \mathbf{f}(t_n + \Delta t, \mathbf{y}_n + \Delta t \mathbf{K}_1), \\ \mathbf{y}_{n+1} &= \mathbf{y}_n + \frac{\Delta t}{2}\left(\mathbf{K}_1 + \mathbf{K}_2\right). \end{aligned} \tag{3.17}$$

It has the Butcher tableau given by

$$\begin{array}{c|cc} 0 & & \\ 1 & 1 & \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array}$$

Heun's method is an explicit method, which is determined by the fact that the values of the $\mathbf{A}$ matrix in the Butcher tableau are zero on and above the diagonal.

3.2.3 The Rush–Larsen method

In 1978, Rush and Larsen [49] proposed a new method for numerically solving cardiac cell models based upon the Hodgkin–Huxley (HH) model of a squid giant axon [22]. Cardiac cell models based upon the HH model involve gating variables that determine the flow of ions through gating channels. The nonlinear ODE for a typical gating variable y is of the form

$$\frac{dy}{dt} = \frac{y_\infty - y}{\tau_y}, \tag{3.18}$$

where

$$y_\infty = \frac{\alpha_y}{\alpha_y + \beta_y} \quad \text{and} \quad \tau_y = \frac{1}{\alpha_y + \beta_y},$$

and where $\alpha_y = \alpha_y(V_m)$ and $\beta_y = \beta_y(V_m)$. The Rush–Larsen (RL) method holds the transmembrane potential V_m constant for each timestep, allowing (3.18) to be treated as a linear ODE with the exact solution

$$y_n = y_\infty + (y_{n-1} - y_\infty)e^{-\frac{\Delta t_n}{\tau_y}}.$$

All other variables of the cell model are solved with the FE method. The RL method is a first-order method and typically more stable than the FE method for cardiac cell models. The increase in stability over the FE method typically permits a larger stable stepsize for a stiff cardiac cell model.

3.2.4 Generalized Rush–Larsen methods

It is possible to extend the RL method, as was done in [60], by applying a local linearization to the RHS of (3.5a) and solving the linearized equations.

GRL1

The RL method is extended into a generalized Rush–Larsen method of order one (GRL1). The GRL1 method decouples and linearizes the ODE system consisting of m ODEs around the point $\mathbf{y} = \mathbf{y}_n$ at time $t = t_n$ to obtain

$$\frac{dy_i}{dt} = f_i(\mathbf{y}_n) + \frac{\partial}{\partial y_i} f_i(\mathbf{y}_n)\,(y_i - y_{n,i}), \quad y_i(t_n) = y_{n,i}, \tag{3.19}$$

for $i = 1, 2, \ldots, m$, where the subscript i denotes component i of a vector. The exact solution of (3.19) is given by

$$y_i(t) = y_{n,i} + \frac{a}{b}\left(e^{b(t-t_n)} - 1\right), \quad i = 1, 2, \ldots, m, \tag{3.20}$$

where $a = f_i(\mathbf{y}_n)$ and $b = \partial f_i(\mathbf{y}_n)/\partial y_i$. The numerical solution $\mathbf{y}_{n+1}$ at time $t = t_{n+1}$ is obtained by

$$y_{n+1,i} = y_{n,i} + \frac{a}{b}\left(e^{b(\Delta t_n)} - 1\right), \quad i = 1, 2, \ldots, m.$$

In practice, if $|\partial f_i(\mathbf{y})/\partial y_i| < \delta$, where $\delta = 10^{-8}$ for double-precision calculations, the limit as $\partial f_i(\mathbf{y})/\partial y_i \to 0$ is used instead of (3.20) to get

$$y_i(t) = y_{n,i} + a(t - t_n), \quad i = 1, 2, \ldots, m.$$

The numerical solution, which is also exact when $\partial f_i(\mathbf{y})/\partial y_i \to 0$, is then obtained by

$$y_{n+1,i} = y_{n,i} + a\Delta t_n, \quad i = 1, 2, \ldots, m.$$

GRL2

The generalized Rush–Larsen method of order two (GRL2) similarly decouples and linearizes the ODE system consisting of m ODEs around the point $\mathbf{y} = \mathbf{y}_n$ at time $t = t_n$ to obtain (3.19) with

exact solution (3.20). However, now the numerical solution $\mathbf{y}_{n+1}$ at time $t = t_{n+1}$ is obtained in two steps:

1. Estimate the solution at time $t_{n+1/2}$ with

$$y_{n+1/2,i} = y_{n,i} + \frac{a}{b}\left(e^{b(\Delta t_n/2)} - 1\right), \quad i = 1, 2, \ldots, m.$$

2. Let $\bar{\mathbf{y}}_{n+1/2} = \mathbf{y}_{n+1/2}$ but with component i replaced by $y_{n,i}$. For each i, compute the numerical solution at time t_{n+1} from

$$y_{n+1,i} = y_{n,i} + \frac{\bar{a}}{\bar{b}}\left(e^{\bar{b}\Delta t_n} - 1\right), \quad i = 1, 2, \ldots, m,$$

where $\bar{a} = f_i(\bar{\mathbf{y}}_{n+1/2})$, $\bar{b} = \partial f_i(\bar{\mathbf{y}}_{n+1/2})/\partial y_i$ and we have used the fact that $\bar{\mathbf{y}}_{n+1/2,i} = y_{n,i}$.

In order to use the generalized RL (GRL) methods, the diagonal of the Jacobian matrix $\partial \mathbf{f}/\partial \mathbf{y}$ is required. When an analytical Jacobian is not available, a special implementation for computing the numerical Jacobian is done in practice because only the diagonal elements are required. This reduces computational cost because unnecessary components of the Jacobian matrix are not computed. The finite-difference approximation of $\partial f_i(\mathbf{y})/\partial y_i$ is obtained by

$$\partial f_i(\mathbf{y})/\partial y_i \approx \frac{f_i(y_1, \ldots, y_{i-1}, y_i + \Delta, y_{i+1}, \ldots, y_m) - f_i(\mathbf{y})}{\Delta},$$

where $\Delta = 10^{-8}$ for double-precision calculations. As was done for the GRL1 method, if $|\partial f_i(\mathbf{y})/\partial y_i| < \delta$ in Step 1, the numerical solution after the first step is obtained by

$$y_{n+1/2,i} = y_{n,i} + a\frac{\Delta t_n}{2}, \quad i = 1, 2, \ldots, m.$$

If $|\partial f_i(\mathbf{y})/\partial y_i| < \delta$ in Step 2, the numerical solution is obtained by

$$y_{n+1,i} = y_{n,i} + \bar{a}\Delta t_n, \quad i = 1, 2, \ldots, m.$$

In theory, the original RL method and the GRL methods solve the gating equations (3.18) with the same algorithm. In practice, however, extra preprocessing is required for the RL method because the gating variables must be identified and treated separately from the remaining variables. The main difference between the methods occurs when dealing with the non-gating variables, which are also treated with an exponential formula based on local linearization in the GRL methods.

Order of convergence

The GRL1 method is now proven to be first order. First, consider the non-autonomous system of m ODEs

$$\frac{d\mathbf{y}}{dt} = \mathbf{f}(t, \mathbf{y}). \tag{3.21}$$

Let $\mathbf{Y}(t) = (\mathbf{y}(t), t)^T$ and let $\bar{\mathbf{Y}}(t) = (\mathbf{Y}_1(t), \cdots, \mathbf{Y}_m(t))^T$. Writing $\mathbf{Y}(t) = \left(\bar{\mathbf{Y}}(t), \mathbf{Y}_{m+1}(t)\right)^T$, we can rewrite (3.21) as an autonomous system of $m+1$ ODEs,

$$\frac{d\mathbf{Y}(t)}{dt} = \mathbf{F}(\mathbf{Y}(t)),$$

where $\mathbf{F}(\mathbf{Y}(t)) = \left(\mathbf{f}(\mathbf{Y}_{m+1}(t), \bar{\mathbf{Y}}(t)), 1\right)^T$. Hence a given non-autonomous system of m ODEs can be rewritten as an autonomous system of $m+1$ ODEs. Now without loss of generality, we can consider the autonomous IVP

$$\frac{d\mathbf{y}}{dt} = \mathbf{f}(\mathbf{y}), \qquad \mathbf{y}(t_n) = \mathbf{y}_n. \tag{3.22}$$

The order of convergence is found by expanding the exact solution to (3.22) in a Taylor series and comparing this expansion to a Taylor series expansion of the numerical solution obtained by the GRL1 method. First, the Taylor series expansion for the solution to (3.22) is given by

$$\mathbf{y}_{n+1} = \mathbf{y}_n + \Delta t \frac{d\mathbf{y}_n}{dt} + O((\Delta t)^2).$$

Using (3.22), we have

$$\mathbf{y}_{n+1} = \mathbf{y}_n + \Delta t f(\mathbf{y}_n) + O((\Delta t)^2).$$

The Taylor series expansion for the solution to the i^{th} component of (3.22) is given by

$$y_{n+1,i} = y_{n,i} + \Delta t f_i(\mathbf{y}_n) + O((\Delta t)^2), \quad i = 1, 2, \ldots, m.$$

Recall that the GRL1 method linearizes (3.22), obtaining for the i^{th} component the equation

$$\frac{d\tilde{y}_i}{dt} = f_i(\mathbf{y}_n) + (\tilde{y}_i - y_{n,i}) \frac{\partial}{\partial y_i} f_i(\mathbf{y}_n), \qquad y_i(t_n) = y_{n,i}, \quad i = 1, 2, \ldots, m. \tag{3.23}$$

The decoupled linear equations (3.23) are solved exactly to $t = t_{n+1}$ for each i. The Taylor series expansion for the i^{th} component solution is given by

$$\tilde{y}_{n+1,i} = y_{n,i} + \Delta t \frac{d\tilde{y}_{n,i}}{dt} + O((\Delta t)^2).$$

Now, using (3.23), we write

$$\tilde{y}_{n+1,i} = y_{n,i} + \Delta t \left(f_i(\mathbf{y}_n) + (\tilde{y}_{n,i} - y_{n,i}) \frac{\partial}{\partial y_i} f_i(\mathbf{y}_n) \right) + O((\Delta t)^2).$$

Using the initial condition, namely $\tilde{y}_{n,i} = y_{n,i}$, we write

$$\tilde{y}_{n+1,i} = y_{n,i} + \Delta t \left(f_i(\mathbf{y}_n) + (y_{n,i} - y_{n,i}) \frac{\partial}{\partial y_i} f_i(\mathbf{y}_n) \right) + O((\Delta t)^2),$$

or

$$\tilde{y}_{n+1,i} = y_{n,i} + \Delta t f_i(\mathbf{y}_n) + O((\Delta t)^2).$$

Now, subtracting the Taylor series expansions for $\tilde{y}_{n+1,i}$ and $y_{n+1,i}$, we obtain

$$\begin{aligned} |\tilde{y}_{n+1,i} - y_{n+1,i}| &= \left(y_{n,i} + \Delta t f_i(\mathbf{y}_n) + O((\Delta t)^2)\right) - \left(y_{n,i} + \Delta t f_i(\mathbf{y}_n) + O((\Delta t)^2)\right), \\ &= O((\Delta t)^2), \end{aligned}$$

for $i = 1, 2, \ldots, m$. Hence we have that the GRL1 method is first order, as required. The GRL2 method is verified to be second order in [2].

3.2.5 Type-insensitive methods

In general, an IVP with an interval of integration from $[t_0, t_f]$ may be stiff for parts of the interval and non-stiff in others [44]. This motivates the use of type-insensitive (TI) methods. A TI method combines a stiff method with a non-stiff method. The stiff method is used for the stiff region(s) of the time interval and the non-stiff method is used for the remainder of the time interval.

Typically type-insensitive solvers analyze the stiffness of a problem at each timestep, e.g., [44]. However, this approach adds to the computational expense of the integration, and the solver may choose the wrong method, further compounding inefficiencies. Because of the periodicity of cardiac simulation at the cellular level, we can pre-determine the regions of stiffness and non-stiffness for each cardiac cell model considered and avoid analyzing the stiffness of the system at each timestep, thus reducing the computational expense of TI methods.

The regions of stiffness and non-stiffness are determined by examining the eigenvalues of the Jacobian matrix of the RHS of the ODE over the interval of interest. The interval of stiffness is taken to be the interval with large negative real eigenvalues, and the non-stiff interval is the remainder of the interval. By taking advantage of this partitioning of the time interval, it is possible for both solvers to use stepsizes governed by accuracy considerations and not stability considerations on their respective intervals, leading to reductions in the overall computational time required to solve the ODE over the entire time interval.

3.3 Numerical solution of PDEs

In order to solve a PDE numerically, both time and space must be discretized. In order to discretize spatially, two methods are commonly used: the Finite Difference Method (FDM) and the Finite Element Method (FEM). Both methods approximate the solution on a set of discrete nodes in a mesh. The complexity of the geometry of the domain determines the complexity of the mesh. For a complex geometry, additional nodes may be required in order to approximate the domain accurately. For the purposes of this section, it is assumed that the nodes in the mesh are uniformly distributed, with equal spacing between each adjacent node. For a more detailed description of the FDM or the FEM, see [7] or [61].

3.3.1 The Finite Difference Method

The FDM method approximates the partial derivatives in the PDE using finite difference approximations. Consider as an example the initial-boundary-value problem (IBVP) consisting of the one-dimensional heat equation with constant coefficient of diffusion D subject to Dirichlet bound-

ary conditions, given by

$$\begin{cases} \dfrac{\partial u}{\partial t} = D\dfrac{\partial^2 u}{\partial x^2}, & a \le x \le b,\ 0 < t \le T, \\ u(a,t) = u_a(t), \\ u(b,t) = u_b(t), \\ u(x,0) = f(x). \end{cases}$$

We divide the domain from a to b uniformly into $M+1$ nodes, $x_0, x_1, x_2, \ldots, x_j, x_{j+1}, \ldots, x_M$, and let $\Delta x = x_{j+1} - x_j$. We divide also the time interval from 0 to T uniformly into $N+1$ nodes, $t_0, t_1, t_2, \ldots, t_i, t_{i+1}, \ldots, t_N$, and let $\Delta t = t_{i+1} - t_i$. Let the solution at the node (t_i, x_j), $u(t_i, x_j)$, be denoted by $u_{i,j}$. We can approximate $\frac{du}{dt}$ for sufficiently small Δt by

$$\frac{du}{dt} = \lim_{\Delta t \to 0} \frac{\Delta u}{\Delta t} \approx \frac{\Delta u}{\Delta t}. \tag{3.24}$$

By using a Taylor series expansion, we can show that the approximation (3.24) has error term $O(\Delta t)$. Writing out the Taylor series expansion for $u(t+\Delta t)$, we have

$$u(t+\Delta t) = u(t) + \Delta t\frac{du}{dt} + \frac{(\Delta t)^2}{2!}\frac{d^2u}{dt^2} + \frac{(\Delta t)^3}{3!}\frac{d^3u}{dt^3} + \cdots$$

Rearranging, we write

$$u(t+\Delta t) - u(t) = \Delta t\frac{du}{dt} + \frac{(\Delta t)^2}{2!}\frac{d^2u}{dt^2} + \frac{(\Delta t)^3}{3!}\frac{d^3u}{dt^3} + \cdots$$

$$\frac{u(t+\Delta t) - u(t)}{\Delta t} = \frac{du}{dt} + \frac{(\Delta t)}{2!}\frac{d^2u}{dt^2} + \frac{(\Delta t)^2}{3!}\frac{d^3u}{dt^3} + \cdots$$

or dropping terms with power of Δt greater than two, we have

$$\frac{du}{dt} = \frac{u(t+\Delta t) - u(t)}{\Delta t} + O(\Delta t) = \frac{\Delta u}{\Delta t} + O(\Delta t).$$

Similarly, it is possible to approximate $\frac{d^2u}{dx^2}$ for small Δx by

$$\frac{d^2u}{dx^2} = \frac{u_{j+1} - 2u_j + u_{j-1}}{(\Delta x)^2} + O((\Delta x)^2).$$

Now, we can rewrite a discretized version of the original IBVP to obtain

$$\begin{cases} \dfrac{u_{i+1,j} - u_{i,j}}{\Delta t} = D\dfrac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{(\Delta x)^2}, & x \in (x_0, x_1, \ldots, x_M),\ t \in (t_0, t_1, \ldots, t_N), \\ u_{i,0} = u_a(t_i), & i = 0, 1, 2, \ldots, N, \\ u_{i,M} = u_b(t_i), & i = 0, 1, 2, \ldots, N, \\ u_{0,j} = f(x_j), & j = 0, 1, 2, \ldots, M. \end{cases}$$

This discretization leads to an update equation for each $u_{i,j}$ in the domain that is dependent on the surrounding values of $u_{i,j}$. Irregularly shaped domains, such as a heart, are difficult to discretize and solve using the FDM. This has made the use of the FEM a popular choice among practitioners.

3.3.2 The Finite Element Method

There are three main steps to follow to solve a PDE using the FEM. First, a weak form of the equations is determined by introducing a function space in which we seek our solution. Second, a discrete subspace of the function space is determined. Finally, the weak form of the PDE is solved over the discrete subspace. Following Chapter 3 of [61], the FEM begins by using the weak form, or the variational formulation, of the PDE to set up an integral form of the problem. A problem is converted into its variational problem by introducing a function space V in which we are seeking the solution to the PDE. The problem is multiplied by arbitrary functions, called test functions, from the function space V, and the problem is integrated over its domain. Consider the following boundary-value problem with domain Ω and boundary $\partial\Omega$ as an example, given by

$$-\nabla^2 u = f(\mathbf{x}), \qquad \mathbf{x} \in \Omega, \tag{3.25}$$

$$u = 0, \qquad \mathbf{x} \in \partial\Omega. \tag{3.26}$$

Let ψ be an arbitrary test function in V. Multiply (3.25) by ψ and integrate to obtain

$$-\int_\Omega \nabla^2 u\, \psi \, d\mathbf{x} = \int_\Omega f(\mathbf{x})\psi \, d\mathbf{x}. \tag{3.27}$$

Now, we have a variational problem, where we need to find a $u \in V$ that satisfies (3.27) for all $\psi \in V$. Using Green's lemma (see, e.g., Chapter 8 of [51]) for the LHS of (3.27), we have

$$\int_\Omega \nabla u \cdot \nabla \psi \, d\mathbf{x} - \int_{\partial\Omega} \hat{\mathbf{n}} \cdot \nabla u \, \psi \, d\mathbf{x} = \int_\Omega f(\mathbf{x})\psi \, d\mathbf{x}, \tag{3.28}$$

where $\hat{\mathbf{n}}$ is the outward unit normal vector. Because the function space V is arbitrary, we can choose V such that all functions $\psi \in V$ are zero on the boundary, reducing (3.28) to

$$\int_\Omega \nabla u \cdot \nabla \psi \, d\mathbf{x} = \int_\Omega f(\mathbf{x})\psi \, d\mathbf{x}, \qquad \forall\, \psi \in V. \tag{3.29}$$

The next step is to discretize (3.29) so that it can be solved numerically. This is done by finding a discrete subspace $V_h \subset V$. One approach is to partition the domain Ω into a set of polygonal sub-domains and let V_h be defined by piecewise polynomial functions over the partitioned sub-domains. In particular, triangles are typically used in 2D and tetrahedra are used in 3D. We define Ω_h as a polygonal approximation to Ω and V_h to be a space of piecewise polynomial functions defined over Ω_h. In particular, we choose piecewise linear functions and let $M+1$ be the number of vertices in Ω_h, with coordinates $\mathbf{x}_i$, $i = 0, 1, 2, \ldots, M$. Then Ω_h is the space spanned by basis functions ϕ_j, $j = 0, 1, 2, \ldots, M$, defined by

$$\Phi_j(\mathbf{x}_i) = \begin{cases} 1, & \text{if } i = j, \\ 0, & \text{otherwise.} \end{cases}$$

Now, we can write (3.29) as a discrete problem

$$\int_{\Omega_h} \nabla u_h \cdot \nabla \psi_h \, d\mathbf{x} = \int_{\Omega_h} f(\mathbf{x})\psi_h \, d\mathbf{x}, \qquad \forall\, \psi_h \in V_h. \tag{3.30}$$

We can write u_h in terms of basis functions

$$u_h = \sum_{j=0}^{M} u_j \phi_j \,,$$

where the u_j are scalars. Using this equation for u_h in (3.30) and using the basis functions as test functions, we have

$$\int_{\Omega_h} \nabla \left(\sum_{j=0}^{M} u_j \phi_j \right) \cdot \nabla \phi_i \, d\mathbf{x} = \int_{\Omega_h} f(\mathbf{x}) \phi_i \, d\mathbf{x}, \qquad i = 0, 1, 2, \ldots, M. \tag{3.31}$$

Because differential operators are linear, we can rewrite (3.31), obtaining

$$\sum_{j=0}^{M} u_j \int_{\Omega_h} \nabla \phi_j \cdot \nabla \phi_i \, d\mathbf{x} = \int_{\Omega_h} f(\mathbf{x}) \phi_i \, d\mathbf{x}, \qquad i = 0, 1, 2, \ldots, M.$$

This approach leads to what is called a Galerkin method because the same basis functions are used as test functions and as the basis with which to approximate the solution. Because the basis functions are known, the integrals are computed, leaving a linear system to be solved for the discrete solution values u_j. In matrix form, the problem becomes

$$\mathbf{A}\mathbf{u} = \mathbf{f},$$

where

$$A_{ij} = \int_{\Omega_h} \nabla \phi_i \cdot \nabla \phi_j \, d\mathbf{x},$$
$$f_i = \int_{\Omega_h} f \phi_i \, d\mathbf{x}.$$

Then the piecewise linear function u_h is the approximate solution to the solution of the continuous problem (3.25).

3.3.3 The FEM for the bidomain model

By using either the semi-implicit method described in Section 3.4.1, ignoring the I_{ion} term, or operator splitting for the bidomain model, we are left with the following PDEs for domain Ω and boundary $\partial\Omega$,

$$\chi C_m \frac{\partial V_m}{\partial t} = \nabla \cdot (\mathbf{M}_I \nabla V_m) + \nabla \cdot (\mathbf{M}_I \nabla u_E) \,, \tag{3.32a}$$
$$0 = \nabla \cdot (\mathbf{M}_I \nabla V_m) + \nabla \cdot ((\mathbf{M}_I + \mathbf{M}_E) \nabla u_E) \,, \tag{3.32b}$$

with boundary conditions

$$\hat{\mathbf{n}} \cdot (\mathbf{M}_I \nabla V_m + \mathbf{M}_I \nabla u_E) = 0, \tag{3.33a}$$
$$\hat{\mathbf{n}} \cdot (\mathbf{M}_E \nabla u_E) = 0. \tag{3.33b}$$

The FEM is used to discretize (3.32). First, let V denote the function space where we are looking for a solution. Next, we multiply (3.32) by a test function $\phi \in V$ and integrate to get

$$\begin{aligned}\int_\Omega \chi C_m \frac{\partial V_m}{\partial t}\phi \, d\mathbf{x} &= \int_\Omega \nabla \cdot (\mathbf{M}_I \nabla V_m)\, \phi \, d\mathbf{x} + \int_\Omega \nabla \cdot (\mathbf{M}_I \nabla u_E)\, \phi \, d\mathbf{x}, \\ 0 &= \int_\Omega \nabla \cdot (\mathbf{M}_I \nabla V_m)\, \phi \, d\mathbf{x} + \int_\Omega \nabla \cdot ((\mathbf{M}_I + \mathbf{M}_E)\, \nabla u_E)\, \phi \, d\mathbf{x}.\end{aligned}$$

Using Green's lemma, we have

$$\begin{aligned}0 &= \frac{d}{dt}\int_\Omega \chi C_m V_m \phi \, d\mathbf{x} + \int_\Omega \mathbf{M}_I \nabla V_m \cdot \nabla \phi \, d\mathbf{x} - \int_{\partial\Omega} \phi \, (\mathbf{M}_I \nabla V_m \cdot \hat{\mathbf{n}}) d\mathbf{s} \\ &+ \int_\Omega \mathbf{M}_I \nabla u_E \cdot \nabla \phi \, d\mathbf{x} - \int_{\partial\Omega} \phi \, (\mathbf{M}_I \nabla u_E \cdot \hat{\mathbf{n}}) d\mathbf{s}, \\ 0 &= \int_\Omega \mathbf{M}_I \nabla V_m \cdot \nabla \phi \, d\mathbf{x} - \int_{\partial\Omega} \phi \, (\mathbf{M}_I \nabla V_m \cdot \hat{\mathbf{n}}) d\mathbf{s} + \\ &\int_\Omega (\mathbf{M}_I + \mathbf{M}_E) \nabla u_E \cdot \nabla \phi \, d\mathbf{x} - \int_{\partial\Omega} \phi \, ((\mathbf{M}_I + \mathbf{M}_E) \nabla u_E \cdot \hat{\mathbf{n}}) d\mathbf{s}.\end{aligned}$$

Using the boundary conditions (3.33a) and (3.33b), we have

$$\begin{aligned}0 &= \frac{d}{dt}\int_\Omega \chi C_m V_m \phi \, d\mathbf{x} + \int_\Omega \mathbf{M}_I \nabla V_m \cdot \nabla \phi \, d\mathbf{x} + \int_\Omega \mathbf{M}_I \nabla u_E \cdot \nabla \phi \, d\mathbf{x}, \\ 0 &= \int_\Omega \mathbf{M}_I \nabla V_m \cdot \nabla \phi \, d\mathbf{x} + \int_\Omega (\mathbf{M}_I + \mathbf{M}_E) \nabla u_E \cdot \nabla \phi \, d\mathbf{x},\end{aligned}$$

for all ϕ in V. Choosing basis functions ψ_j, $j = 0, 1, 2, \ldots, M$, for our domain Ω, we let $V_m \approx \sum_{j=0}^{M} v_j \psi_j$ and $u_E \approx \sum_{j=0}^{M} u_j \psi_j$. Now, we get discretized equations for (3.32),

$$\begin{aligned}0 =& \chi C_m \frac{d}{dt}\sum_{j=0}^{M} v_j \int_\Omega \psi_j \, \psi_i \, d\mathbf{x} + \sum_{j=0}^{M} v_j \int_\Omega \mathbf{M}_I \nabla \psi_j \cdot \nabla \psi_i \, d\mathbf{x} + \sum_{j=0}^{M} u_j \int_\Omega \mathbf{M}_I \nabla \psi_j \cdot \nabla \psi_i \, d\mathbf{x}, \\ 0 =& \sum_{j=0}^{M} v_j \int_\Omega \mathbf{M}_I \nabla \psi_j \cdot \nabla \psi_i \, d\mathbf{x} + \sum_{j=0}^{M} u_j \int_\Omega (\mathbf{M}_I + \mathbf{M}_E) \nabla \psi_j \cdot \nabla \psi_i \, d\mathbf{x},\end{aligned}$$

for $i = 0, 1, 2, \ldots, M$. The derivatives in time remain to be discretized, but this can be done using any numerical method for ODEs.

3.3.4 Error norms for PDEs

In order to compare the accuracy and efficiency of numerical methods for solving a system of PDEs to a given error tolerance, it is necessary to have a measure of the accuracy of the numerical method used. Assuming an IBVP is solved for $t \in [t_0, t_f]$ over a domain Ω, this can be done by computing an average of the error at $N+1$ temporal points in $[t_0, t_f]$ and for $M+1$ spatial points in Ω. The MRMS error is generalized from (3.3) by

$$e_{\text{MRMS}} = \sqrt{\frac{1}{NM}\sum_{i=0}^{N}\sum_{j=0}^{M}\left(\frac{\hat{y}_{i,j} - y_{i,j}}{1 + |\hat{y}_{i,j}|}\right)^2}, \tag{3.34}$$

where $y_{i,j}$ is the numerical solution and $\hat{y}_{i,j}$ is the reference solution at time t_i and spatial location x_j, $i = 0, 1, 2, \ldots, N$, $j = 0, 1, 2, \ldots, M$.

The RRMS error is generalized from (3.4) by

$$e_{\text{RRMS}} = \sqrt{\frac{1}{NM}\frac{\sum_{i=0}^{N}\sum_{j=0}^{M}(\hat{y}_{i,j}-y_{i,j})^2}{\sum_{i=0}^{N}\sum_{j=0}^{M}(\hat{y}_{i,j})^2}}, \tag{3.35}$$

where $y_{i,j}$ is the numerical solution and $\hat{y}_{i,j}$ is the reference solution at time t_i and spatial location x_j, $i = 0, 1, 2, \ldots, N$, $j = 0, 1, 2, \ldots, M$.

3.4 Methods used in the software

Because we are interested in improving the efficiency of the numerical methods used to solve the bidomain model, it is important to understand how the available software packages are currently solving the bidomain model. We consider only CARP, PROPAG, Continuity 6, and Chaste because PyCC is no longer actively maintained and the software by Ying et al. [70] solves only the monodomain model. CARP uses operator splitting to decouple the bidomain model into a parabolic PDE, an elliptic PDE, and a system of nonlinear ODEs. For simple problems, all three are solved with the first-order Forward Euler method. For problems that require a finer mesh, the second-order Crank–Nicolson method is used to solve the parabolic PDE. PROPAG decouples the bidomain model into a system of ODEs and two decoupled PDEs. The ODEs are solved first in order to update the I_{ion} term, required to solve for the transmembrane potential in an explicit discretization of the first PDE. The extracellular potential is updated by solving a linear system of equations resulting from the discretization of the second PDE [47]. Continuity 6 uses operator splitting, with the ODEs solved using RADAU5 [28]. Work was done in [28] to use Graphics processing units (GPUs) within Continuity 6 to solve the system of ODEs using the Backward Euler method with a single iteration of the Newton–Raphson method to solve the nonlinear system, greatly reducing run-time. Chaste uses a semi-implicit first-order method to solve the bidomain model, described in Section 3.4.1. This reduces the problem to solving a linear system of equations and a system of ODEs. By default, the Forward Euler method is used to solve the ODEs but it is possible to use other methods including the Backward Euler method and Heun's method.

3.4.1 Semi-implicit method for cardiac modelling

A semi-implicit time discretization method, as described in [43], is used in Chaste to solve the bidomain model and represents an alternative to applying operator splitting to the full bidomain model. By using a semi-implicit method, the stability restriction on stepsize is reduced and a linear system of equations and a system of nonlinear ODEs are solved instead of a larger, nonlinear system of equations, occurring from the discretization of the complete bidomain model. The PDEs (2.1b) and (2.1c) can be discretized semi-implicitly as follows, using timestep Δt, and the notation $V_m(\mathbf{x}, n\Delta t) = V_{m,n}$, $u_E(\mathbf{x}, n\Delta t) = u_{E,n}$, and $\mathbf{s}(\mathbf{x}, n\Delta t) = \mathbf{s}_n$, to obtain

$$\chi C_m \frac{V_{m,n+1} - V_{m,n}}{\Delta t} + \chi \mathbf{I}_{ion}(\mathbf{s}_n, V_{m,n}, t) = \nabla \cdot (\mathbf{M}_I \nabla V_{m,n+1}) + \nabla \cdot (\mathbf{M}_I \nabla u_{E,n+1}),$$
$$0 = \nabla \cdot (\mathbf{M}_I \nabla V_{m,n+1}) + \nabla \cdot ((\mathbf{M}_I + \mathbf{M}_E) \nabla u_{E,n+1}).$$

In Chaste, the FEM is used for spatial discretization of the PDEs. Following the spatial discretization described in Section 3.3.3, we are left with a linear system of the form $\mathbf{Ax} = \mathbf{b}$ to solve at each timestep. Assuming we have basis functions ψ_j for our domain Ω, the linear system is given by

$$\begin{pmatrix} \frac{\chi C_m}{\Delta t}\mathbf{M} + \mathbf{A}_I & \mathbf{A}_I \\ \mathbf{A}_I & \mathbf{A}_I + \mathbf{A}_E \end{pmatrix} \begin{pmatrix} \mathbf{V}_{m,n+1} \\ \mathbf{u}_{E,n+1} \end{pmatrix} = \begin{pmatrix} \frac{\chi C_m}{\Delta t}\mathbf{M}\mathbf{V}_{m,n} - \frac{1}{C_m}\mathbf{I}_{ion}(\mathbf{s}_n, \mathbf{V}_{m,n}) \\ \mathbf{0} \end{pmatrix}, \quad (3.36)$$

where $\mathbf{M} = \int_\Omega \psi_j \, \psi_i \, d\mathbf{x}$, $\mathbf{A}_I = \int_\Omega \mathbf{M}_I \nabla \psi_j \cdot \nabla \psi_i \, d\mathbf{x}$, and $\mathbf{A}_E = \int_\Omega \mathbf{M}_E \nabla \psi_j \cdot \nabla \psi_i \, d\mathbf{x}$, for $\psi_j, \psi_i \in \Omega$.

For $t_n < t < t_n + \Delta t$, the steps to solve the bidomain model are given by

1. Using a numerical method for IVPs, solve the system of nonlinear ODEs at each node, given by
$$\frac{d\mathbf{s}}{dt} = \mathbf{f}(\mathbf{s}, V_m, t),$$
where $\mathbf{f}$ is determined by the cell model.

2. Set up the RHS of the linear system (3.36).

3. Solve the linear system (3.36).

3.4.2 Contributions to Chaste

In order to run the experiments for this thesis, additions were made to Chaste. At the time of writing, the latest release version is Chaste 3.0, available since January 2012. The following is a list of additions and contributions made to Chaste. For each addition incorporated into Chaste, the version number of the software is also mentioned.

1. The second-order Heun's method (3.17) for solving ODEs has been added. It has been present in Chaste since version 2.0, which was released in April 2010. When computing reference solutions for most of the 1D bidomain simulations described in Section 4.4, this method was used to solve the ODEs because it achieved sufficient convergence more quickly than the FE method or the BE method in Chaste.

2. The ability to output solutions with any number of digits has been added. A refactored version of this addition is present in Chaste 3.0. The contributed version was modified by

the main Chaste developers to better fit with the software. Previously, the number of digits output to the simulation solution file was limited to six or fewer digits. This addition was done to facilitate the comparison of numerical solutions to determine the number of matching digits between two numerical solutions.

3. The Rush–Larsen method for the Luo–Rudy model, described in Section 3.2.3, has been added and enabled the Chaste developers to add the Rush–Larsen method to PyCML. With this addition, Rush–Larsen type code can be automatically generated for all compatible cell models available from CellML. This addition is present in Chaste 3.0.

4. The GRL1 and GRL2 methods, described in Section 3.2.4, have been added to PyCML. With this addition, GRL1 and GRL2 code can be automatically generated for all compatible cell models available from CellML. This addition is expected to be present in the next release of Chaste (after July 2012).

Chapter 4

Results

This chapter presents the cardiac cell models used in this study and explains why they are of interest in Section 4.1. The RRMS and the MRMS error norms are compared for the model of McAllister et al. (1975) in Section 4.2 and the reliability of the MRMS error norm is shown. The results from the study of 37 cardiac cell models and the application of the results from the cell model study to a one-dimensional bidomain problem are reported in Sections 4.3 and 4.4, respectively. The cardiac cell study in this thesis extends the study reported in [56] and includes the corrected results from [57]. The additions to the cardiac cell study from [56] include a comparison using the MRMS error norm and the addition of the GRL1 method.

4.1 Cardiac cell models

4.1.1 Cardiac cell models used

Cardiac cell models have been developed for a range of mammals such as rats, mice, guinea pigs, rabbits, dogs, and humans. Mammals other than humans are studied because their hearts have similar properties and are easier to study. Models also exist for specific cardiac cell types such as atrial, ventricular, Purkinje fibre, and sino-atrial cardiac cells. The combination of the quantity of mammals studied and the number of heart cell types leads to a vast selection of available cell models.

This thesis focuses on 37 cardiac cell models that are derived from the HH model of a squid giant axon. The 37 cell models range from the smallest, the FHN model, with two variables, to the largest, the model of Bondarenko et al. (2004), with 41 variables. The FHN model is a simplification of the HH model and was rescaled and modified to have realistic values for the action potential, following [61]. The models were chosen to represent a wide sampling of the cell models available, in terms of species, cell type, and model detail. They were also selected based on their CellML rating. Only models marked with a gold star on the CellML website that have been verified to match with published results were considered, with the exception of the model of Puglisi–Bers (2001). For the model of Puglisi–Bers (2001), other verified code was available from [56]. A complete list of the cell models chosen, together with the number of variables for each model, a reference to the original

paper, and a brief description of the model, are reported in Table 4.1. The models include the LR model of a guinea pig ventricular cell, the Winslow et al. (1999) model of a canine ventricular cell, and the Courtemanche et al. (1998) model of a human atrial cell.

Note that the model of Winslow et al. (1999) used is a reduced form of the original model, with 31 variables, and is referred to by Winslow31 throughout the remainder of this thesis. The intracellular sodium concentration and one of the calcium handing mechanisms from the original model are set as constant [62]. Because the endocardial, epicardial, and M-cell variants of the models of Sakmann et al. (2000), ten Tusscher et al. (2004), and ten Tusscher et al. (2006) were available as separate CellML files from the CellML repository, all three of the variants are included in this study.

For each model, an electrical stimulus is applied in order to initiate an action potential. The maximum stepsize allowed for each model is restricted to the duration of time in which the stimulus current was applied. Stepsizes that have reached this maximum stepsize are marked by a dagger. Note also that stepsizes were adjusted to land exactly on the beginning and end points of the application of the stimulus current and on the end point of the simulation. This was done to avoid introducing errors caused by using a discontinuous function for the stimulus current.

4.1.2 Analysis of the eigenvalues of the cardiac cell models

Numerical eigenvalues for the Jacobian of the RHS of the system of ODEs for all 37 cell models were computed at one ms time intervals in order to determine the stiffness of each cell model. The numerical eigenvalues were computed using Matlab's `eig` function to evaluate the Jacobian at each interval. The Jacobian matrix was computed using Matlab's `numjac` function and verified using ADiMat, an automatic differentiation software package for Matlab [5]. Table 4.2 reports the extreme values for the real and imaginary components of the eigenvalues over the time interval and the percentage of simulation time when a complex eigenvalue pair was present. By considering the maximum magnitude of the negative real eigenvalues from Table 4.2, the five stiffest cell models in order of decreasing stiffness are identified as those of Pandit et al. (2003), Winslow31, Bondarenko et al. (2004), Pandit et al. (2001), and Jafri et al. (1998).

By examining the eigenvalues of a cell model, it is possible to determine the regions of stiffness that can be exploited by the TI methods. For example, consider the model of Pandit et al. (2001) with 26 variables. Figure 4.1 shows the solution solved over 250 ms for the transmembrane potential and Figure 4.2 shows the maximum and minimum real part of the eigenvalues over the interval of integration. By examining Figure 4.2, the interval $t \in [105\ 125]$ can be considered to be the stiff region and the union of the intervals $t \in [0\ 105]$, $t \in [125\ 250]$ can be considered to be the non-stiff region. The TI method uses the BE, RL, GRL1, or GRL2 method for the stiff region and the FE method for the non-stiff region.

Table 4.1: Summary of the 37 cardiac cell models used in this thesis. Three types of cardiac cell variants (endocardial cell, epicardial cell, and M-cell) exist for each of the models marked with an asterisk.

Model	Reference	Number of variables	Description
Beeler–Reuter (1977)	[4]	8	Mammalian ventricular model
Bondarenko et al. (2004)	[6]	41	Mouse ventricular model
Courtemanche et al. (1998)	[10]	21	Human atrial model
Demir et al. (1994)	[12]	27	Rabbit sinoatrial node model
Demir et al. (1999)	[11]	29	Rabbit sinoatrial node model
DiFrancesco–Noble (1985)	[14]	16	Mammal Purkinje fibre model
Dokos et al. (1996)	[15]	18	Rabbit sinoatrial node model
Faber–Rudy (2000)	[16]	19	Guinea pig ventricular model
FitzHugh–Nagumo (1961)	[17, 34]	2	Nerve membrane model
Fox et al. (2002)	[18]	13	Canine ventricular model
Hilgemann–Noble (1987)	[21]	15	Rabbit atrial model
Hund–Rudy (2004)	[23]	29	Canine ventricular model
Jafri et al. (1998)	[25]	31	Guinea pig ventricular model
Luo–Rudy (1991)	[30]	8	Guinea pig ventricular model
Maleckar et al. (2008)	[31]	30	Human atrial model
McAllister et al. (1975)	[33]	10	Canine Purkinje fibre model
Noble (1962)	[36]	4	Mammal Purkinje fibre model
Noble–Noble (1984)	[37]	15	Rabbit sinoatrial node model
Noble et al. (1991)	[38]	17	Guinea pig ventricular model
Noble et al. (1998)	[39]	22	Guinea pig ventricular model
Nygren et al. (1998)	[40]	29	Human atrial model
Pandit et al. (2001)	[41]	26	Rat left-ventricular model
Pandit et al. (2003)	[42]	26	Rat left-ventricular model
Puglisi–Bers (2001)	[48]	17	Rabbit ventricular model
Sakmann et al. (2000)*	[50]	21	Guinea pig ventricular model
Stewart et al. (2009)	[58]	20	Human Purkinje fibre model
Ten Tusscher et al. (2004)*	[64]	17	Human ventricular model
Ten Tusscher et al. (2006)*	[65]	19	Human ventricular model
Wang–Sobie (2008)	[68]	35	Neonatal mouse ventricular model
Winslow31	[69]	31	Canine ventricular model
Zhang et al. (2000)	[72]	15	Rabbit sinoatrial node model

Table 4.2: Extreme values of the eigenvalues for each cell model. The minimum real part of the set of eigenvalues is denoted $\min(Re(\lambda))$ and the maximum real part of the set of eigenvalues is denoted $\max(Re(\lambda))$. Similarly, the minimum and maximum imaginary parts are denoted $\min(Im(\lambda))$ and $\max(Im(\lambda))$. The percentage of the solution interval in which there is at least one pair of complex eigenvalues is also reported.

Model	$\min(Re(\lambda))$	$\max(Re(\lambda))$	$\min(Im(\lambda))$	$\max(Im(\lambda))$	% Complex
Beeler–Reuter (1977)	–8.20E+1	1.55E–2	–1.97E+0	1.97E+0	45
Bondarenko (2004)	–8.49E+3	4.51E+0	–2.80E+0	2.80E+0	53
Courtemanche et al. (1998)	–1.29E+2	1.87E–1	–4.50E+0	4.50E+0	82
Demir et al. (1994)	–3.80E+1	4.79E–1	–7.95E–2	7.95E–2	74
Demir et al. (1999)	–3.82E+1	4.81E–1	–7.95E–2	7.95E–2	72
DiFrancesco–Noble (1985)	–2.63E+1	1.88E+0	–6.14E–1	6.14E–1	56
Dokos et al. (1996)	–2.99E+1	5.06E–1	–1.19E–1	1.19E–1	97
Faber–Rudy (2000)	–1.84E+2	1.37E–2	–5.61E–1	5.61E–1	58
FitzHugh–Nagumo (1961)	–4.39E–1	1.78E–1	–4.59E–2	4.59E–2	28
Fox et al. (2002)	–4.39E+2	4.44E–2	–4.19E–1	4.19E–1	65
Hilgemann–Noble (1987)	–3.25E+1	1.58E–1	–2.25E–1	2.25E–1	25
Hund–Rudy (2004)	–1.95E+2	9.22E–1	–3.74E+0	3.74E+0	62
Jafri et al. (1998)	–4.42E+3	4.82E+0	–2.35E–1	2.35E–1	47
Luo–Rudy (1991)	–1.51E+2	7.01E–2	–4.11E–2	4.11E–2	73
Maleckar et al. (2008)	–4.16E+1	2.42E–1	–3.43E–1	3.43E–1	28
McAllister et al. (1975)	–1.83E+2	1.49E+0	–3.02E+0	3.02E+0	68
Noble (1962)	–9.80E+0	1.74E+0	–1.28E–1	1.28E–1	24
Noble–Noble (1984)	–1.25E+1	4.77E–1	–1.03E–1	1.03E–1	92
Noble et al. (1991)	–3.89E+1	4.35E+0	–1.72E–1	1.72E–1	20
Noble et al. (1998)	–3.60E+1	5.71E+0	–2.35E–1	2.35E–1	47
Nygren et al. (1998)	–4.03E+1	2.05E+0	–3.88E–1	3.88E–1	24
Pandit et al. (2001)	–6.92E+3	4.30E+0	–1.43E+0	1.43E+0	12
Pandit et al. (2003)	–7.54E+4	3.87E+0	–9.11E–1	9.11E–1	35
Puglisi–Bers (2001)	–1.91E+2	2.22E+0	–1.07E–1	1.07E–1	41
Sakmann et al. (2000) – Endo	–2.97E+1	7.21E–1	–7.48E–2	7.48E–2	84
Sakmann et al. (2000) – Epi	–2.96E+1	6.98E–1	–7.47E–2	7.47E–2	75
Sakmann et al. (2000) – M-cell	–2.98E+1	1.98E+0	–7.58E–2	7.58E–2	72
Stewart et al. (2009)	–1.38E–1	3.34E–3	–1.57E–3	1.57E–3	92
Ten Tusscher et al. (2004) – Endo	–1.17E+3	1.01E–1	–4.64E+0	4.64E+0	17
Ten Tusscher et al. (2004) – Epi	–1.17E+3	9.74E–2	–4.70E+0	4.70E+0	18
Ten Tusscher et al. (2004) – M-cell	–1.17E+3	9.75E–2	–4.70E+0	4.70E+0	21
Ten Tusscher et al. (2006) – Endo	–1.26E+3	4.00E+0	–4.77E+0	–4.77E+0	50
Ten Tusscher et al. (2006) – Epi	–9.44E+2	2.84E+0	–5.01E+0	5.01E+0	51
Ten Tusscher et al. (2006) – M-cell	–9.81E+2	4.36E+0	–4.64E+0	4.64E+0	34
Wang–Sobie (2008)	–1.23E+2	1.23E+0	–1.24E+0	1.24E+0	46
Winslow31	–1.84E+4	1.53E+0	–4.22E–1	4.22E–1	63
Zhang et al. (2000)	–2.22E+1	1.29E–1	–1.00E–1	1.00E–1	89

A similar analysis to determine the region of stiffness was also applied to the models of Bondarenko et al. (2004), Jafri et al. (1998), and Winslow31. Stiff and non-stiff intervals are listed for these four models in Table 4.3.

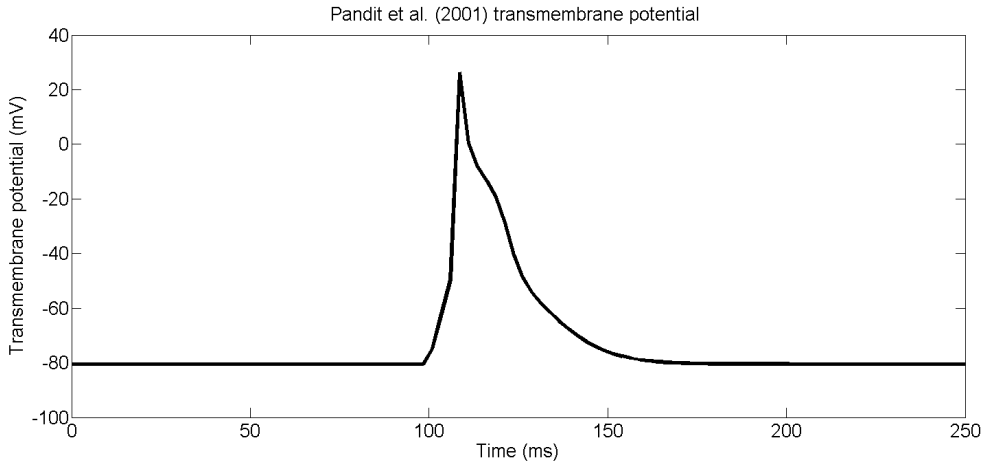

Figure 4.1: Transmembrane potential for the model of Pandit et al. (2001).

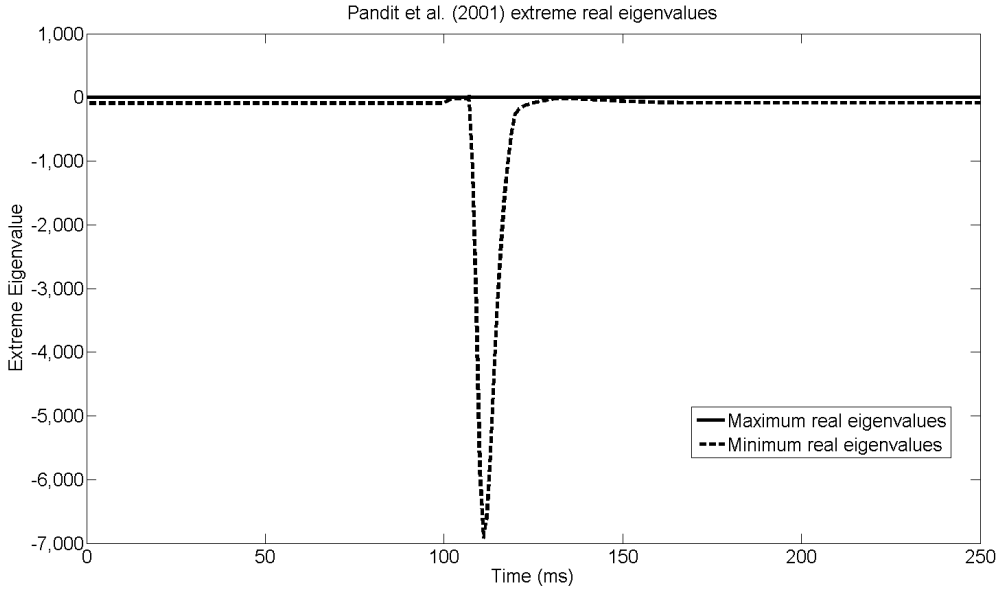

Figure 4.2: Extreme real eigenvalue values for the model of Pandit et al. (2001).

4.2 Error norm comparison

The RRMS and MRMS error norms are compared at 5% and 1% error for the model of McAllister et al. (1975), solved using the RL method. Figure 4.3 compares a reference solution correct to seven

Table 4.3: Stiffness intervals for four models.

Model	Stiff interval	Non-stiff interval
Bondarenko et al. (2004)	[20 30]	[0 20]; [30 75]
Jafri et al. (1998)	[0 50]	[50 300]
Pandit et al. (2001)	[105 125]	[0 105]; [125 250]
Winslow31	[0 50]	[50 300]

matching digits (see Section 4.3 for a description of the computation of the reference solutions) to solutions computed to 5% and 1% RRMS error for the transmembrane potential. It can be seen that at 5% RRMS error, the action potential is early by approximately 100 ms. The solution computed to 1% RRMS error is fairly accurate compared to the reference solution. Figure 4.4 compares the same reference solution to solutions computed to 5% and 1% MRMS error for the transmembrane potential for the sub-interval 200–250 ms. It can be seen that at 1% MRMS error, the solution is extremely accurate. At 5% MRMS error, the solution remains accurate but is eight times faster to compute.

From examination of Figures 4.3 and 4.4, we postulate that for the purpose of clinical accuracy requirements, the RRMS error norm at 5% is too relaxed and the MRMS error norm at 1% is too strict. Similar results hold for the RRMS and MRMS error norms for the remaining 36 cell models. Therefore, for the remainder of this chapter, only results for the MRMS error norm at 5% are shown. The results for the RRMS error norm at 5% and 1% and the results for the MRMS error norm at 1% can be found in Appendix A.

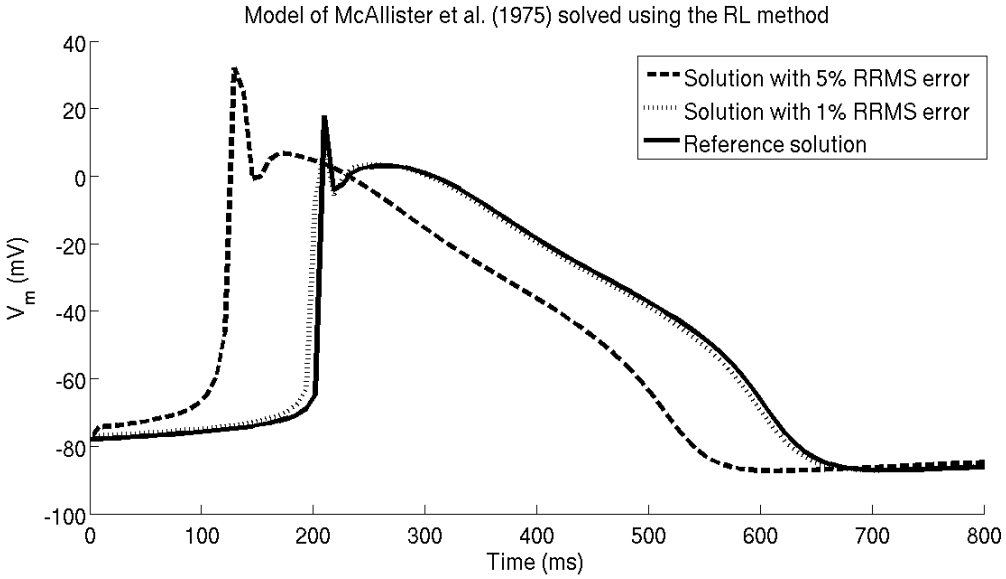

Figure 4.3: RRMS error at 5% and 1% for the model of McAllister et al. (1975) solved using the RL method.

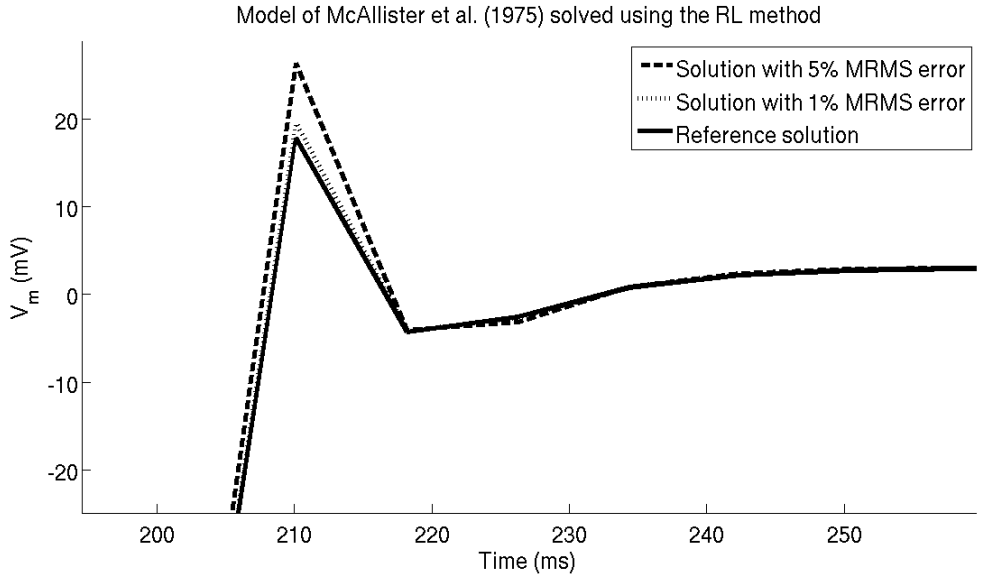

Figure 4.4: MRMS error at 5% and 1% for the model of McAllister et al. (1975) solved using the RL method for the interval [200 250].

4.3 Cardiac cell model results

The FE, RL, GRL1, and GRL2 methods are compared for the 37 cell models described in Section 4.1, with results listed in Table 4.4. The shortest execution time has been highlighted in bold text for each model. The BE method is compared to the FE, RL, GRL1, and GRL2 methods for six models, those of Courtemanche et al. (1998), Faber–Rudy (2000), Noble (1962), Noble et al. (1998), Pandit et al. (2003), and Winslow31, with results listed in Table 4.5. These models were chosen to represent a range of stiffness, from the non-stiff model of Noble (1962), to the moderately stiff model of Courtemanche et al. (1998), to the stiff model of Pandit et al. (2003), in order to test the BE method on a range of models, from non-stiff to moderately stiff to stiff. In Table 4.6, the most efficient method is given out of the FE, RL, GRL1 and GRL2 methods for the five stiffest models at a 5% MRMS error tolerance.

The TI methods are tested on four of the stiffest models, those of Bondarenko et al. (2004), Jafri et al. (1998), Pandit et al. (2001), and Winslow31, with results listed in Table 4.7. The shortest execution time has been highlighted in bold text for each model. The TI methods are tested on the stiffest models because such models are the most likely to show computation time improvements through the use of TI methods. In Table 4.8, the optimal single method is compared to the optimal TI method and the winning method is highlighted in bold text.

By comparing solutions computed with decreasing absolute and relative tolerances, down to 10^{-12}, Matlab's `ode15s` method was used to find reference solutions for all 37 cell models with seven to ten matching digits at $N = 100$ equally spaced points in the interval of integration. The error between the reference solution and the computed solution was computed at $N = 100$ equally spaced points, using first- or second-order interpolation as necessary. Maximum constant stepsizes were found for every method and model combination that gave RRMS and MRMS errors of 5% and 1%. Timings reported are the minimum run time out of 100 runs for the maximum stepsize that satisfied the error tolerance. Timings were computed in Matlab R2010a on an HP Z400 with an Intel Xeon W3520 2.66 GHz quad-core processor with 16 GB of DDR3 RAM running 64-bit Ubuntu 9.04. Hyperthreading and turbo-boost were enabled while the timings were computed.

From the results listed in Table 4.4, the FE method wins for seven models, the RL method method wins for 25 models, the GRL1 method wins for two models, and the GRL2 method wins for three models. From Table 4.5, the BE method is never the most efficient method to compute a solution for the six cell models considered.

It is worth noting that the GRL methods are generally able to solve the stiff models in a shorter time than the other methods considered. With the exception of the Winslow31 model, the GRL methods consistently have the shortest execution time. The results for the five stiffest models are summarized in Table 4.6.

Table 4.4: Stepsize, in milliseconds, and execution time, in seconds, of the four numerical methods using the largest stepsize that produced less than 5% MRMS error. The shortest execution time has been highlighted in bold text for each model.

Model	FE		RL		GRL1		GRL2	
	Δt	Time	Δt	Time	Δt	Time	Δt	Time
Beeler–Reuler (1977)	2.53E–2	4.31E–2	7.20E–1	**1.40E–3**	8.08E–1	3.85E–3	7.15E–1	8.33E–3
Bondarenko et al. (2004)	2.13E–4	2.63E+0	2.13E–4	2.28E+0	7.47E–3	**8.41E–1**	6.92E–3	1.80E+0
Courtemanche et al. (1998)	1.94E–2	2.35E–1	7.97E–2	**5.60E–2**	9.60E–2	3.01E–1	2.98E–1	1.75E–1
Demir et al. (1994)	5.95E–2	1.84E–2	5.32E–2	**1.76E–2**	1.18E–1	9.03E–2	5.41E–1	4.14E–2
Demir et al. (1999)	5.96E–2	**1.95E–2**	4.73E–2	2.75E–2	9.99E–2	1.26E–1	5.32E–1	5.00E–2
DiFrancesco–Noble (1985)	7.73E–2	9.77E–2	1.95E–1	**3.40E–2**	2.07E–1	3.22E–1	7.82E–1	1.78E–1
Dokos et al. (1996)	7.02E–2	3.28E–2	1.22E–1	**1.64E–2**	8.02E–2	2.78E–1	6.72E–1	7.69E–2
FitzHugh–Nagumo (1961)	2.72E–3	**6.05E–2**	NA	NA	2.60E–3	1.35E–1	2.61E–3	2.29E–1
Faber–Rudy (2000)	1.12E–2	2.38E–1	2.01E–2	**1.17E–1**	4.06E–2	6.28E–1	2.04E–1	2.60E–1
Fox et al. (2002)	4.62E–3	3.52E–1	4.33E–2	**3.31E–2**	1.16E–1	8.77E–2	1.31E–1	1.61E–1
Hilgemann–Noble (1987)	6.25E–2	2.47E–2	8.06E–2	**1.51E–2**	1.52E–1	9.77E–2	6.24E–1	5.04E–2
Hund–Rudy (2004)	7.80E–3	**3.61E–1**	5.33E–3	4.85E–1	5.47E–3	4.88E+0	6.74E–2	7.96E–1
Jafri et al. (1998)	5.76E–4	4.20E+0	5.77E–4	**3.59E+0**	1.41E–3	1.71E+1	1.00E–2	4.84E+0
Luo–Rudy (1991)	1.35E–2	1.47E–1	1.23E–1	1.30E–2	3.15E–1	**1.01E–2**	5.95E–1	2.19E–2
Maleckar et al. (2008)	5.02E–2	9.25E–2	8.87E–2	**4.60E–2**	4.20E–1	1.29E–1	9.71E–1	1.14E–1
McAllister et al. (1975)	2.47E–2	9.18E–2	4.69E–1	**4.41E–3**	2.53E–1	2.38E–2	2.43E–1	9.36E–2
Noble (1962)	2.02E–1	4.25E–3	1.47E–1	**3.69E–3**	1.10E–1	1.77E–2	3.11E–1	1.17E–2
Noble–Noble (1984)	2.04E–1	**6.77E–3**	1.21E–1	9.57E–3	9.27E–2	1.21E–1	9.81E–1	2.41E–2
Noble et al. (1991)	5.15E–2	2.56E–2	1.53E–1	**7.46E–3**	1.04E–1	1.17E–1	3.93E–1	6.19E–2
Noble et al. (1998)	5.56E–2	6.22E–2	1.57E–1	**1.96E–2**	8.86E–2	3.47E–1	3.66E–1	1.74E–1
Nygren et al. (1998)	5.36E–2	1.10E–1	8.88E–2	**5.88E–2**	2.06E–1	2.77E–1	5.33E–1	2.28E–1
Pandit et al. (2001)	2.91E–4	5.90E+0	2.91E–4	5.13E+0	2.40E–2	6.02E–1	9.58E–2	**3.01E–1**
Pandit et al. (2003)	2.65E–5	6.34E+1	2.65E–5	5.68E+1	1.57E–2	9.67E–1	3.57E–2	**8.21E–1**
Puglisi–Bers (2001)	5.97E–3	1.94E+0	1.45E–2	7.81E–1	3.23E–2	1.04E+0	2.14E–1	**3.32E–1**
Sakmann et al. (2000) – Endo	6.90E–2	**5.84E–2**	4.99E–2	6.94E–2	4.16E–2	8.87E–1	1.75E–1	4.35E–1
Sakmann et al. (2000) – Epi	6.90E–2	**5.82E–2**	4.16E–2	8.32E–2	3.83E–2	9.67E–1	1.72E–1	4.45E–1
Sakmann et al. (2000) – M-cell	6.86E–2	5.92E–2	2.32E–1	**1.51E–2**	4.21E–1	8.80E–2	1.29E+0	5.97E–2
Stewart et al. (2009)	1.52E+1	5.27E–1	2.05E+2	**3.48E–2**	1.74E+2	3.78E–1	4.20E+2	3.09E–1
Ten Tusscher et al. (2004) –Endo	1.78E–3	2.14E+0	1.24E–1	**2.65E–2**	1.37E–1	2.18E–1	3.44E–1	1.71E–1
Ten Tusscher et al. (2006) –Endo	1.62E–3	1.55E+0	7.03E–2	**3.10E–2**	1.29E–1	1.67E–1	2.16E–1	1.97E–1
Ten Tusscher et al. (2004) –Epi	1.78E–3	2.12E+0	1.12E–1	**2.97E–2**	1.19E–1	2.51E–1	3.21E–1	1.83E–1
Ten Tusscher et al. (2006) –Epi	2.14E–3	1.20E+0	1.16E–1	**1.90E–2**	1.75E–1	1.23E–1	3.04E–1	1.41E–1
Ten Tusscher et al. (2004) –M-cell	1.76E–3	1.58E+0	1.21E–1	**2.03E–2**	1.02E–1	2.23E–1	5.29E–1	8.24E–2
Ten Tusscher et al. (2006) –M-cell	2.06E–3	1.22E+0	1.27E–1	**1.72E–2**	1.38E–1	1.54E–1	3.08E–1	1.38E–1
Wang–Sobie (2008)	1.66E–2	6.88E–2	5.27E–2	**1.90E–2**	9.36E–2	1.20E–1	4.13E–1	5.37E–2
Winslow31	1.07E–4	**1.65E+1**	1.07E–4	1.81E+1	9.38E–5	2.15E+2	7.15E–4	5.94E+1
Zhang et al. (2000)	9.97E–2	5.83E–2	4.57E–1	**1.16E–2**	3.04E–1	1.12E–1	1.41E+0	4.84E–2

Table 4.5: Stepsize, in milliseconds, and execution time, in seconds, of the Backward Euler method using the largest stepsize that produced less than 5% MRMS error. Note that none of the execution times have been highlighted in bold text because the BE method does not have the shortest execution time for any model compared to the FE, RL, GRL1, and GRL2 methods.

Model	BE	
	Δt	Time
Courtemanche et al. (1998)	2.30E–1	4.59E+0
Faber–Rudy (2000)	3.55E–2	1.25E+1
Noble (1962)	2.07E–1	9.50E–1
Noble et al. (1998)	2.83E–1	2.92E+0
Pandit et al. (2003)	3.53E–2	9.97E+0
Winslow31	3.86E–3	1.02E+2

Table 4.6: Most efficient method out of the FE, RL, GRL1 and GRL2 methods for the five stiffest models at a 5% MRMS error tolerance.

Model	Optimal Method
Bondarenko et al. (2004)	GRL1
Jafri et al. (1998)	RL
Pandit et al. (2001)	GRL2
Pandit et al. (2003)	GRL2
Winslow31	FE

Table 4.7: Stiffness intervals and execution time, in seconds, of type-insensitive methods using the largest stepsize that produced less than 5% MRMS error. The shortest execution time has been highlighted in bold text for each model.

Model	Stiff Interval	Non-stiff Interval	RL-FE	GRL1-FE	GRL2-FE	BE-FE
			Time			
Bondarenko et al. (2004)	[20 30]	[0 20]; [30 75]	7.51E–1	**5.16E–1**	6.70E–1	5.97E+0
Jafri et al. (1998)	[0 50]	[50 300]	**1.07E+0**	3.03E+0	1.11E+0	4.81E+0
Pandit et al. (2001)	[105 125]	[0 105]; [125 250]	9.26E+0	1.04E–1	**1.01E–1**	5.48E–1
Winslow31	[0 50]	[50 300]	**3.09E+0**	3.85E+1	1.02E+1	1.93E+1

From the TI method results listed in Table 4.7, the RL-FE combination wins for two models, the GRL1-FE combination wins for one model, and the GRL2-FE combination wins for one model. Table 4.8 compares the TI methods to the most efficient single method and shows that for each model considered, the TI method is more efficient than any single method considered. For the Winslow31 model, the RL-FE combination is five times faster than the FE method, the next most efficient method. For the model of Pandit et al. (2001), the GRL2-FE combination is three times faster than the GRL2 method, the next most efficient method. It is worth noting that, with the exception of the model of Winslow31, the most efficient TI method is the combination method using the most efficient single method as the stiff solver, as is shown in Table 4.8.

The following observations can be made from the results. The FE method is only effective for non-stiff models, and these are usually the least physically realistic. The BE method is not effective for any of the six models considered. Because the BE method is tested for a range of non-stiff to moderately stiff to stiff models, it is not likely to be effective for the remaining 31 models. The RL method is the method of choice for moderately stiff models. From the eigenvalue analysis, the majority of the 37 cell models considered are non-stiff to moderately stiff, explaining the effectiveness of the RL method. The GRL methods are effective for solving stiff models, with the exception of the Winslow31 model. The GRL methods offer significant reductions in computation time, particularly for the models of Pandit et al. (2001) and Pandit et al. (2003). The TI methods

Table 4.8: Most efficient method out of the FE, RL, GRL1, GRL2, and TI methods, using the largest stepsize that produced less than 5% MRMS error. The optimal method has been highlighted in bold text for each model.

Model	Optimal Single Method	Optimal TI method
Bondarenko et al. (2004)	GRL1	**GRL1-FE**
Jafri et al. (1998)	RL	**RL-FE**
Pandit et al. (2001)	GRL2	**GRL2-FE**
Winslow31	FE	**RL-FE**

always outperform the single methods for the four stiff models considered. The additional effort required to identify the regions of stiffness and non-stiffness may be worthwhile depending on the application.

4.4 1D bidomain results

Also of interest is to establish whether the methods effective at solving the system of ODEs remain effective when used to solve the system of ODEs within the bidomain model. The FE, RL, GRL1, and GRL2 methods are compared for their ability to efficiently solve the system of ODEs within a 1D bidomain scenario for eight of the 37 cardiac cell models described in Section 4.1. The eight cell models were chosen because they are compatible with the PyCML addition in Chaste and represent a range of stiffness, from the non-stiff FHN model to the stiff Winslow31 model. The cell models, together with their time interval of integration for the 1D simulation, are listed in Table 4.9. The bidomain model was solved for a one cm spatial interval with the initial condition

$$\begin{aligned} \mathbf{V}_m(t=0,\mathbf{x}) &= \mathbf{V}_{m,0} + 100(1-\sin(\mathbf{x})), \\ \mathbf{s}(t=0,\mathbf{x}) &= \mathbf{s}_0, \\ \mathbf{u}_E(t=0,\mathbf{x}) &= 0, \end{aligned}$$

where $\mathbf{V}_{m,0}$ and $\mathbf{s}_0$ are the default resting state values for $\mathbf{V}_m$ and $\mathbf{s}$, respectively, for the particular cell model used. The parameter values used for the bidomain simulations are given in Table 4.10.

By comparing solutions computed by halving the timestep and doubling the number of mesh points, reference solutions were computed for each 1D bidomain scenario with four or more matching digits. Solutions were compared at 21 equally spaced points in the temporal interval and 101 equally spaced points in the spatial interval, for a total of 2121 points. For all the bidomain simulations

Table 4.9: Cell models used within the 1D bidomain problem, listed with their interval of integration.

Model	Time Interval (ms)
Courtemanche et al. (1998)	[0 5]
FitzHugh–Nagumo (1961)	[0 5]
Fox et al. (2002)	[0 10]
Luo–Rudy (1991)	[0 5]
Maleckar et al. (2008)	[0 10]
Nygren et al. (1998)	[0 9]
Pandit et al. (2003)	[0 10]
Winslow31	[0 10]

Table 4.10: Parameter values used in Chaste to solve the 1D bidomain simulations.

Parameter	Value
C_m	1 μF/cm^2
χ	1400 1/cm
M_I	1.75 mS/cm
M_E	7 mS/cm

described in this thesis, the linear system (3.36) described in Section 3.4.1 was solved using the MUltifrontal Massively Parallel sparse direct Solver (MUMPS) [1]. By default in Chaste, the linear system (3.36) is solved using a preconditioner together with an iterative solver. However, it was found that the iterative solver could not achieve sufficient convergence for the purposes of computing reference solutions.

Maximum constant stepsizes were found that gave MRMS errors of 5%. Because the timesteps for each simulation were constant, they were required to land exactly on the end point of the simulation time interval and on the points over the time interval that were used for computing the error of the solution; consequently somewhat larger constant stepsizes that would meet the 5% MRMS error criterion are likely possible. Results are reported in Table 4.11 and the shortest execution time has been highlighted in bold text for each model. Timings reported are the minimum run time out of 100 runs for the maximum stepsize that satisfied the error tolerance. Timings were computed in Chaste 2.2, built with the GNU compilers with optimized flags, on an HP Z400 with an Intel Xeon W3520 2.66 GHz quad-core processor with 16 GB of DDR3 RAM running 64-bit Ubuntu 9.04. Hyperthreading and turbo-boost were enabled while the timings were computed.

Table 4.11: Timesteps, in milliseconds, and execution time, in seconds, for the FE, RL, GRL1, and GRL2 methods used to solve the ODEs within a 1D bidomain simulation, using the largest timestep that produced less than 5% MRMS error. Timesteps reported are for the ODE and PDE timesteps. The shortest execution time has been highlighted in bold text for each model.

Model		FE		RL		GRL1		GRL2	
	Δx	Δt	Time	Δt	Time	Δt	Time	Δt	Time
Courtemanche et al. (1998)	1.00E–2	2.60E–3	1.07E+0	6.76E–3	**4.68E–1**	6.76E–3	7.69E–1	6.76E–3	1.34E+0
FitzHugh–Nagumo (1961)	1.00E–2	2.50E–1	**4.53E–2**	NA	NA	2.50E–1	4.64E–2	2.50E–1	**4.53E–2**
Fox et al. (2002)	2.00E–2	1.10E–3	6.18E+0	3.18E–3	**2.32E+0**	3.18E–3	3.45E+0	3.18E–3	5.73E+0
Luo–Rudy (1991)	2.00E–2	1.52E–3	1.35E+0	3.57E–3	**6.72E–1**	3.57E–3	8.76E–1	3.57E–3	1.28E+0
Maleckar et al. (2008)	1.00E–2	1.25E–3	3.48E+0	4.55E–3	**1.15E+0**	4.55E–3	2.04E+0	4.67E–3	3.63E+0
Nygren et al. (1998)	1.00E–2	3.49E–4	1.35E+1	1.33E–3	**3.89E+0**	1.33E–3	7.99E+0	1.33E–3	1.58E+1
Pandit et al. (2003)	1.00E–2	2.59E–5	2.01E+2	2.59E–5	2.19E+2	2.27E–3	**4.33E+0**	2.30E–3	7.62E+0
Winslow31	3.00E–2	1.24E–4	9.86E+1	1.24E–4	1.06E+2	9.46E–4	**2.63E+1**	1.29E–3	3.58E+1

From Table 4.11, we have that for the FE, RL, GRL1, and GRL2 comparison for the eight cell models within a 1D bidomain simulation, the FE method and the GRL2 method tie for one model (the FHN model), the RL method wins for five models, and the GRL1 method wins for two models. With the exception of the FHN model, the FE method and the GRL2 method are the most inefficient methods for each model. For the moderately stiff models, the RL, GRL1, and GRL2 methods are able to use approximately the same stepsize. Because the RL method is the least computationally expensive of these three methods per timestep, it is able to compute a solution within the specified error tolerance in the least amount of time. The RL method is 1.3–2.1 times faster than the GRL1 method, the next fastest method. The RL and GRL1 methods are 2–3.5 and 1.4–1.7 times faster than the FE method, respectively. For the two stiff models, that of Pandit et al. (2003) and Winslow31, the GRL methods are able to take much larger timesteps without being restricted by stability. The GRL1 method solves the model of Pandit et al. (2003) 46 and 50 times faster than the FE method and the RL method, respectively, and solves Winslow31 3.7 and 4 times faster than the FE method and the RL method, respectively.

Table 4.12 compares the cell model results to the 1D bidomain simulation results. The cell models that were most efficiently solved by the RL and FE methods are also most efficiently solved by the RL and FE methods within the 1D bidomain simulation. The most efficient methods for the model of Luo–Rudy (1991) solved on its own and within the 1D bidomain simulation are the GRL1 method and the RL method, respectively. From Table 4.4, the cell model timings for the GRL1 and RL methods are within 30% of each other, so the change from the GRL1 method to the RL method as the fastest method for the model of Luo–Rudy (1999) within a 1D bidomain simulation is not completely surprising. The most significant difference is that the GRL1 method is the fastest method for the two stiffest cell models within the 1D bidomain simulation, the model of Pandit et al. (2003) and the Winslow31 model. However, this is not unexpected behaviour because the GRL1 and GRL2 methods are able to take similar stepsizes within the 1D bidomain simulation for

these models, and the GRL1 method is less computationally expensive per step.

Overall, we can conclude that the cell model results transfer well to the 1D bidomain simulation results. The RL method is the most efficient method for the moderately stiff models, the FE method is only efficient for the FHN model, and the GRL methods are the most efficient methods for the two stiffest models. From the analysis of the eigenvalues of the Jacobian matrix computed for each of the 37 cell models considered, it was found that the majority of the cell models range from non-stiff to moderately stiff, with only five significantly stiff models. The pronounced number of moderately stiff models explains why the RL method was the most successful method for the majority of the cardiac cell models solved by a single method and within the one-dimensional bidomain simulations.

Table 4.12: Comparison of the most efficient methods for eight cell models solved independently and within the 1D bidomain model simulation.

Model	Fastest Method	
	Cell Model Only	1D Bidomain Simulation
Courtemanche et al. (1998)	RL	RL
FitzHugh–Nagumo (1961)	FE	FE, GRL2
Fox et al. (2002)	RL	RL
Luo–Rudy (1991)	GRL1	RL
Maleckar et al. (2008)	RL	RL
Nygren et al. (1998)	RL	RL
Pandit et al. (2003)	GRL2	GRL1
Winslow31	FE	GRL1

Chapter 5

Conclusions and Future Work

The heart performs one of the most important functions in the human body, namely that of pumping blood to transport oxygen and to remove waste products from cells. Due to the heart's location and function, it is difficult to study without harming the patient. An alternative to studying a living heart is to mathematically model its electrical and mechanical activity. The bidomain model, coupled with a cardiac cell model that describes the electrical activity of a single heart cell, describes the propagation of the electrical activity throughout a heart. Real-time simulation has not been achieved to date because numerically solving the bidomain model is difficult and computationally demanding. The goal of this thesis was to study methods used to solve the cardiac cell models in order to reduce the computational time necessary to solve the bidomain model.

This thesis focused on the methods used to solve the system of ODEs from the cardiac cell models. A new error norm, the MRMS error norm, was presented and shown to be more reliable for determining error compared to the RRMS error norm. The eigenvalues of 37 cardiac cell models were analyzed and a study of the FE, RL, GRL1, GRL2, BE, and type-insensitive methods used to solve the 37 cardiac cell models was completed. The results from this study were applied to a one-dimensional bidomain problem for eight cell models to determine to what degree the ODE results apply to a bidomain simulation.

From the analysis of the eigenvalues of the 37 cell models considered, it was found that the majority of the cell models are non-stiff to moderately stiff with only a few significantly stiff models. At 5% MRMS error, the RL method was the most efficient for 25 of the 37 cell models and for five of the eight cell models used within the 1D bidomain simulation. Based on the investigation undertaken in this thesis, the RL method remains the optimal choice for moderately stiff cell models, solved independently or solved within the bidomain model.

At 5% MRMS error, the GRL methods were efficient for the five stiffest cell models. The GRL methods were able to take stepsizes determined by accuracy, not stability, for the models of Pandit et al. (2001) and Pandit et al. (2003). The GRL2 method was 17 and 69 times faster than the RL method for the models of Pandit et al. (2001) and Pandit et al. (2003), respectively. Within the 1D bidomain simulation, the GRL1 method was the most efficient for the two stiffest models, the model of Pandit et al. (2003) and the Winslow31 model. The GRL1 method was 46 and four times

faster than the RL method for the models of Pandit et al. (2003) and Winslow31, respectively.

From the results listed in Chapter 4, some general conclusions follow. First and foremost, from Table 4.12 it can be seen that the results of the study of the ODEs from the cardiac cell models can be applied to the one-dimensional bidomain problem. From Tables 4.4, 4.5, and 4.11, the FE method is efficient only for the non-stiff ODE systems such as the FHN model and the BE method is slower than the fastest of the FE, RL, GRL1, and GRL2 methods for each of the six models considered, making the FE and BE methods the least effective methods for solving cardiac cell models. From Table 4.7, type-insensitive methods are shown to be effective for solving stiff problems. The TI methods are 1.6 to 5.3 times faster than single methods for the four models considered.

From the results listed in Chapter 4, some general recommendations follow. It is recommended that the RL method be used to solve the moderately stiff cell models and the GRL1 or GRL2 methods be used to solve the stiff cell models. Within a 1D bidomain simulation, it is recommended that the GRL1 method be used because it is competitive with the RL method for the moderately stiff models and it is more efficient for the stiff models. The Rush–Larsen method and the generalized Rush–Larsen method of order one are able to reduce the computational time required to solve a one-dimensional problem by 2–46 times compared to the de facto standard, the FE method.

This thesis concludes with two unanswered questions that could lead to possible future work:

1. In Section 4.1.2, the eigenvalues of the Jacobian matrix of the RHS of cardiac cell models were approximated over time. By examining the magnitude of the negative real parts of these eigenvalues, the stiffness of the cardiac cell models was determined and regions of stiffness and non-stiffness were determined for the stiff models. Of the 37 models studied, most were found to be moderately stiff, and five were found to be highly stiff. By examining the eigenvectors associated with the real eigenvalues having large negative parts, it may be possible to determine which variables in the cell models are causing the stiffness. If it is possible to ascertain which of the variables in a stiff model are stiff, it would be interesting to investigate whether the TI methods can be improved by applying the stiff solver to only the stiff variables over the stiff interval of integration.

2. In Chapter 4, the FE, RL, GRL1, and GRL2 methods were used to solve the ODEs from eight cardiac cell models within a 1D bidomain simulation, and timings were compared for each method. From the results listed in Table 4.12 in Section 4.4, the ODE results were found to be relevant to one-dimensional bidomain simulations. It would be interesting to investigate whether the ODE results and the one-dimensional bidomain simulation results are also relevant to two- and three-dimensional bidomain simulations in order to reduce the computational time required to solve the simulations to a given accuracy.

References

[1] MUMPS: A parallel sparse direct solver. http://graal.ens-lyon.fr/MUMPS/index.php?page=home, January 2012.

[2] R. Artebrant, J. Sundnes, O. Skavhaug, and A. Tveito. A second order method of Rush–Larsen type. Technical report, Simula Research Laboratory, 2008.

[3] K.E. Atkinson, W. Han, and D.E. Stewart. *Numerical solution of ordinary differential equations*. John Wiley & Sons, Inc., Hoboken, New Jersey, 2009.

[4] G.W. Beeler and H. Reuter. Reconstruction of the action potential of ventricular myocardial fibres. *J. Physiol.*, 268(1):177–210, 1977.

[5] C. H. Bischof, H. M. Bücker, and A. Vehreschild. A macro language for derivative definition in ADiMat. In *Automatic differentiation: applications, theory, and implementations*, volume 50 of *Lect. Notes Comput. Sci. Eng.*, pages 181–188. Springer, Berlin, 2006.

[6] V.E. Bondarenko, G.P. Szigeti, G.C.L. Bett, S.-J. Kim, and R.L. Rasmusson. Computer model of action potential of mouse ventricular myocytes. *Am. J. Physiol. Heart Circ. Physiol.*, 287(3):H1378–H1403, 2004.

[7] B. Bradie. *A friendly introduction to numerical analysis*. Pearson Prentice Hall, Upper Saddle River, NJ, 2006.

[8] R.L. Burden and J.D. Faires. *Numerical analysis*. Thomson Brooks/Cole, 2005.

[9] Statistics Canada. Statistics Canada: Leading causes of death. http://www.statcan.gc.ca/daily-quotidien/101130/dq101130b-eng.htm, November 2011.

[10] M. Courtemanche, R.J. Ramirez, and S. Nattel. Ionic mechanisms underlying human atrial action potential properties: insights from a mathematical model. *Am. J. Physiol.*, 275(1):H301–H321, 1998.

[11] S.S. Demir, J.W. Clark, and W.R. Giles. Parasympathetic modulation of sinoatrial node pacemaker activity in rabbit heart: a unifying model. *Am. J. Physiol.*, 276(6 Pt 2):H2221–H2244, 1999.

[12] S.S. Demir, J.W. Clark, C.R. Murphey, and W.R. Giles. A mathematical model of a rabbit sinoatrial node cell. *Am. J. Physiol.*, 266(3 Pt 1):C832–C852, 1994.

[13] S. Descombes. Convergence of a splitting method of high order for reaction-diffusion systems. *Mathematics of computation*, 70(236):1481–1501, 2001.

[14] D. DiFrancesco and D. Noble. A model of cardiac electrical activity incorporating ionic pumps and concentration changes. *Philos. Trans. R. Soc. Lond. B Biol. Sci.*, 307(1133):353–398, 1985.

[15] S. Dokos, B. Celler, and N. Lovell. Ion currents underlying sinoatrial node pacemaker activity: a new single cell mathematical model. *J. Theor. Biol.*, 181(3):245–272, 1996.

[16] G.M. Faber and Y. Rudy. Action potential and contractility changes in [Na(+)](i) overloaded cardiac myocytes: a simulation study. *Biophys. J.*, 78(5):2392–2404, 2000.

[17] R. FitzHugh. Impulses and Physiological States in Theoretical Models of Nerve Membrane. *Biophys. J.*, 1(6):445–466, 1961.

[18] J.J. Fox, J.L. McHarg, and R.F. Gilmour, Jr. Ionic mechanism of electrical alternans. *Am. J. Physiol. Heart Circ. Physiol.*, 282(2):H516–H530, 2002.

[19] S.K. Godunov. A difference method for numerical calculation of discontinuous solutions of the equations of hydrodynamics. *Math. Sbornik*, 47(89):271–306, 1959.

[20] The Cardiac Mechanics Research Group. Continuity 6: A problem solving environment for multi-scale biology. http://www.continuity.ucsd.edu/Continuity/, November 2011.

[21] D.W. Hilgemann and D. Noble. Excitation-contraction coupling and extracellular calcium transients in rabbit atrium: reconstruction of basic cellular mechanisms. *Proc. R. Soc. Lond. B Biol. Sci.*, 230(1259):163–205, 1987.

[22] A.L. Hodgkin and A.F. Huxley. A quantitative description of membrane current and its application to conduction and excitation in nerve. *J. Physiol. (Lond)*, 117(4):500–544, 1952.

[23] T.J. Hund and Y. Rudy. Rate dependence and regulation of action potential and calcium transient in a canine cardiac ventricular cell model. *Circulation*, 110(20):3168–3174, 2004.

[24] Auckland Bioengineering Institute. The CellML project. http://www.cellml.org/, November 2011.

[25] M.S. Jafri, J.J. Rice, and R.L. Winslow. Cardiac Ca2+ dynamics: the roles of ryanodine receptor adaptation and sarcoplasmic reticulum load. *Biophys. J.*, 74(3):1149–1168, 1998.

[26] A. M. Katz. *Physiology of the heart*. Lippincott Williams and Wilkins, Philadelphia, PA, 2006.

[27] D. Lanser and J. G. Verwer. Analysis of operator splitting for advection-diffusion-reaction problems from air pollution modelling. *Journal of Computational and Applied Mathematics*, 111:201–216, 1999.

[28] F. Lionetti. GPU accelerated cardiac electrophysiology. Master's thesis, University of California, San Diego, 2010.

[29] C. M. Lloyd, M. D. B. Halstead, and P. F. Nielsen. CellML: its future, present and past. *Progress in Biophysics and Molecular Biology.*, 85:433–450, 2004.

[30] C. Luo and Y. Rudy. A model of ventricular cardiac action potential. *Circ. Res.*, 68(6):1501–1526, 1991.

[31] M.M. Maleckar, J.L. Greenstein, N.A. Trayanova, and W.R. Giles. Mathematical simulations of ligand-gated and cell-type specific effects on the action potential of human atrium. *Prog. Biophys. Mol. Biol.*, 98(2-3):161–170, 2008.

[32] MATLAB. *version 7.12.0 (R2011a)*. The MathWorks Inc., Natick, Massachusetts, 2011.

[33] R.E. McAllister, D. Noble, and R.W. Tsien. Reconstruction of the electrical activity of cardiac Purkinje fibres. *J. Physiol.*, 251(1):1–59, 1975.

[34] J. Nagumo, S. Arimoto, and S. Yoshizawa. An active pulse transmission line simulating nerve axon. *Proceedings of the IRE*, 50(10):2061–2070, 1962.

[35] S. Niederer, L. Mitchell, N. Smith, and G. Plank. Simulating human cardiac electrophysiology on clinical time-scales. *Frontiers in Physiology*, 2(14):1–7, 2011.

[36] D. Noble. A modification of the Hodgkin–Huxley equations applicable to Purkinje fibre action and pace-maker potentials. *J. Physiol.*, 160:317–352, 1962.

[37] D. Noble and S.J. Noble. A model of sino-atrial node electrical activity based on a modification of the DiFrancesco-Noble (1984) equations. *Proc. R. Soc. Lond. B Biol. Sci.*, 222(1228):295–304, 1984.

[38] D. Noble, S.J. Noble, G.C. Bett, Y.E. Earm, W.K. Ho, and I.K. So. The role of sodium-calcium exchange during the cardiac action potential. *Ann. N. Y. Acad. Sci.*, 639:334–353, 1991.

[39] D. Noble, A. Varghese, P. Kohl, and P. Noble. Improved guinea-pig ventricular cell model incorporating a diadic space, IKr and IKs, and length- and tension-dependent processes. *Can. J. Cardiol.*, 14(1):123–134, 1998.

[40] A. Nygren, C. Fiset, L. Firek, J.W. Clark, D.S. Lindblad, R.B. Clark, and W.R. Giles. Mathematical model of an adult human atrial cell: the role of K+ currents in repolarization. *Circ. Res.*, 82(1):63–81, 1998.

[41] S.V. Pandit, R.B. Clark, W.R. Giles, and S.S. Demir. A mathematical model of action potential heterogeneity in adult rat left ventricular myocytes. *Biophys. J.*, 81(6):3029–3051, 2001.

[42] S.V. Pandit, W.R. Giles, and S.S. Demir. A mathematical model of the electrophysiological alterations in rat ventricular myocytes in type-I diabetes. *Biophys. J.*, 84(2 Pt 1):832–841, 2003.

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

[44] L. Petzold. Automatic selection of methods for solving stiff and nonstiff systems of ordinary differential equations. *SIAM J. Sci. Statist. Comput.*, 4(1):137–148, 1983.

[45] 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. Rodriguez, 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.

[46] M. Potse. Propag. http://www.potse.nl/project/, September 2011.

[47] M. Potse, B. Dube, 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 Trans. Biomed. Eng.*, 53(12):2425–2435, 2006.

[48] J.L. Puglisi and D.M. Bers. Labheart: an interactive computer model of rabbit ventricular myocyte ion channels and Ca transport. *Am. J. Physiol. Cell Physiol.*, 281(6):C2049–C2060, 2001.

[49] S. Rush and H. Larsen. A practical algorithm for solving dynamic membrane equations. *IEEE Trans. Biomed. Eng.*, BME-25(4):389–392, 1978.

[50] B.F. Sakmann, A.J. Spindler, S.M. Bryant, K.W. Linz, and D. Noble. Distribution of a persistent sodium current across the ventricular wall in guinea pigs. *Circ. Res.*, 87(10):910–914, 2000.

[51] T. Sauer. *Numerical analysis*. Pearson Education, Inc., Boston, MA, 2006.

[52] L.F. Shampine, I. Gladwell, and S. Thompson. *Solving ODEs with MATLAB*. Cambridge University Press, Cambridge, U.K., 2003.

[53] Q. Sheng. Solving linear partial differential equations by exponential splitting. *IMA Journal of Numerical Analysis*, 9:199–212, 1989.

[54] O. Skavhaug. The python heart solver (pycc). http://folk.uio.no/skavhaug/heart_simulations.html, September 2011.

[55] A.T. Sornborger. Higher-order operator splitting methods for deterministic parabolic equations. *International Journal of Computer Mathematics*, 84(6):887–893, 2007.

[56] R. J. Spiteri and R. C. Dean. Stiffness analysis of cardiac electrophysiological models. *Annals of Biomedical Engineering*, 38(12):3592–3604, 2010.

[57] R. J. Spiteri and M. E. Marsh. Erratum to: Stiffness analysis of cardiac electrophysiological models. *Annals of Biomedical Engineering*, Under review.

[58] P. Stewart, O.V. Aslanidi, D. Noble, P.J. Noble, M.R. Boyett, and H. Zhang. Mathematical models of the electrical action potential of Purkinje fibre cells. *Philos. Transact. A Math. Phys. Eng. Sci.*, 367(1896):2225–2255, 2009.

[59] G. Strang. On the construction and comparison of difference schemes. *SIAM J. Numer. Anal.*, 5(3), 1968.

[60] J. Sundnes, R. Artebrant, O. Skavhaug, and A. Tveito. A second-order algorithm for solving dynamic cell membrane equations. *IEEE Trans. Biomed. Eng.*, 56(10):2546–2548, 2009.

[61] J. Sundnes, G. T. Lines, X. Cai, B. F. Nielsen, K. A. Mardal, and A. Tveito. *Computing the electrical activity in the heart*. Springer-Verlag, Berlin, 2006.

[62] J. Sundnes, G. T. Lines, and A. Tveito. Efficient solution of ordinary differential equations modeling electrical activity in cardiac cells. *Mathematical Biosciences*, 172(2):55–72, 2001.

[63] J. Sundnes, G. T. Lines, and A. Tveito. An operator splitting method for solving the bidomain equations coupled to a volume conductor model for the torso. *Mathematical Biosciences*, 194:233–248, 2005.

[64] 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(4):H1573–H1589, 2004.

[65] 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(3):H1088–H1100, 2006.

[66] L. Tung. *A bi-domain model for describing ischemic myocardial dc potentials*. PhD thesis, Massachusetts Institute of Technology, 1978.

[67] E.D. Vigmond and G. Plank. Carp. http://carp.meduni-graz.at/, September 2011.

[68] L.J. Wang and E.A. Sobie. Mathematical model of the neonatal mouse ventricular action potential. *Am. J. Physiol. Heart. Circ. Physiol.*, 294(6):H2565–H2575, 2008.

[69] 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(5):571–586, 1999.

[70] W. Ying, D.J. Rose, and C.S. Henriquez. Efficient fully implicit time integration methods for modeling cardiac dynamics. *IEEE Trans. Biomed. Eng.*, 55(12):2701–2711, 2008.

[71] H. Yoshida. Construction of higher order symplectic integrators. *Physics Letters A*, 150(5,6,7):262–268, November 1990.

[72] H. Zhang, A.V. Holden, I. Kodama, H. Honjo, M. Lei, T. Varghese, and M.R. Boyett. Mathematical models of action potentials in the periphery and center of the rabbit sinoatrial node. *Am. J. Physiol. Heart Circ. Physiol.*, 279(1):H397–H421, 2000.

Appendix A

Additional ODE Results

Additional results were computed for the 37 cardiac cell models using the RRMS error norm at 5% and 1% and the MRMS error norm at 1%. However, because it was shown that the RRMS error norm at 5% and the MRMS error norm at 1% are not good measures of the accuracy of a solution, the results have been omitted from Chapter 4 and included here for completeness. The RRMS error norm at 1% gives similar results to the MRMS error norm at 5%. In each table of results, the shortest execution time has been highlighted in bold text for each model. Note that none of the execution times have been highlighted in bold text for the BE method results because the BE method does not have the shortest execution time for any model compared to the FE, RL, GRL1, and GRL2 methods for the RRMS error norm at 5% and 1% RRMS or for the MRMS error norm at 1%. Note that the maximum stepsize allowed for each model is restricted to the duration of time in which the stimulus current was applied. Stepsizes that have reached this maximum stepsize are marked by a dagger.

Table A.1: Stepsize, in milliseconds, and execution time, in seconds, of the four numerical methods using the largest stepsize that produced less than 5% RRMS error. The shortest execution time has been highlighted in bold text for each model. Stepsizes that have reached the maximum stepsize are marked by a dagger.

Model	FE		RL		GRL1		GRL2	
	Δt	Time	Δt	Time	Δt	Time	Δt	Time
Beeler–Reuter (1977)	2.53E–2	4.39E–2	1.00E+0†	**1.02E–3**	1.00E+0†	3.10E–3	1.00E+0†	5.90E–3
Bondarenko et al. (2004)	2.13E–4	2.64E+0	2.13E–4	2.30E+0	7.48E–3	8.51E–1	2.85E–2	**4.44E–1**
Courtemanche et al. (1998)	1.94E–2	2.35E–1	2.00E+0†	**2.25E–3**	2.00E+0†	1.47E–2	2.00E+0†	2.64E–2
Demir et al. (1994)	5.95E–2	1.82E–2	1.53E–1	6.19E–3	2.99E+0	3.68E–3	7.30E+0	**3.30E–3**
Demir et al. (1999)	5.98E–2	1.93E–2	1.53E–1	8.69E–3	2.99E+0	4.29E–3	7.30E+0	**3.79E–3**
DiFrancesco–Noble (1985)	7.92E–2	9.57E–2	2.65E+1	3.33E–4	1.50E+3	**1.54E–4**	1.50E+3	1.92E–4
Dokos et al. (1996)	7.02E–2	3.33E–2	3.33E+0	**6.80E–4**	6.17E+0	3.75E–3	1.48E+1	3.74E–3
Faber–Rudy (2000)	1.12E–2	2.45E–1	5.00E–1†	**4.79E–3**	5.00E–1†	5.10E–2	5.00E–1†	1.09E–1
FitzHugh–Nagumo (1961)	5.00E–1†	**3.73E–4**	N/A	N/A	5.00E–1†	8.53E–4	5.00E–1†	1.26E–3
Fox et al. (2002)	4.62E–3	3.53E–1	1.00E+0†	**1.49E–3**	1.00E+0†	9.99E–3	1.00E+0†	2.12E–2
Hilgemann–Noble (1987)	6.25E–2	2.49E–2	8.06E–2	1.51E–2	6.00E+0	**2.68E–3**	7.31E+0	4.49E–3
Hund–Rudy (2004)	1.11E–2	2.50E–1	1.90E–1	**1.37E–2**	2.53E–1	1.06E–1	3.89E–1	1.40E–1
Jafri et al. (1998)	5.76E–4	4.17E+0	5.33E–4	3.88E+0	3.21E–2	**7.46E–1**	1.03E–2	4.71E+0
Luo–Rudy (1991)	1.35E–2	1.50E–1	4.37E–1	4.13E–3	1.00E+0†	**3.18E–3**	1.00E+0†	1.36E–2
Maleckar et al. (2008)	5.02E–2	9.49E–2	8.87E–2	4.60E–2	5.04E+0	**1.08E–2**	6.00E+0†	1.85E–2
McAllister et al. (1975)	2.76E–2	8.33E–2	4.50E+0	5.23E–4	4.31E+1	**2.20E–4**	2.19E+1	1.16E–3
Noble (1962)	2.12E–1	3.93E–3	2.05E+0	**3.23E–4**	1.59E+0	1.53E–3	3.93E+0	1.01E–3
Noble–Noble (1984)	2.04E–1	6.66E–3	9.72E+0	**2.02E–4**	1.33E+1	9.82E–4	3.23E+1	8.96E–4
Noble et al. (1991)	5.15E–2	2.57E–2	1.53E–1	7.46E–3	1.77E+0	**7.07E–3**	1.84E+0	1.37E–2
Noble et al. (1998)	5.58E–2	6.03E–2	1.57E–1	1.97E–2	2.47E+0	**1.27E–2**	2.76E+0	2.32E–2
Nygren et al. (1998)	5.36E–2	1.11E–1	8.88E–2	5.87E–2	5.00E+0†	**1.14E–2**	5.00E+0†	2.45E–2
Pandit et al. (2001)	2.91E–4	5.90E+0	2.91E–4	5.13E+0	1.08E–1	**1.34E–1**	9.58E–2	3.02E–1
Pandit et al. (2003)	2.65E–5	6.34E+1	2.65E–5	5.68E+1	2.05E–1	**7.56E–2**	1.96E–1	1.49E–1
Puglisi–Bers (2001)	1.08E–1	1.00E+0	4.99E–1	**2.23E–2**	7.14E–1	4.61E–2	7.14E–1	9.86E–2
Sakmann et al. (2000) – Endo	6.90E–2	5.84E–2	2.36E–1	1.48E–2	3.00E+0†	**1.25E–2**	3.00E+0†	2.58E–2
Sakmann et al. (2000) – Epi	6.90E–2	5.89E–2	2.36E–1	1.48E–2	3.00E+0†	**1.24E–2**	3.00E+0†	2.58E–2
Sakmann et al. (2000) – M-cell	6.86E–2	5.87E–2	2.36E–1	1.48E–2	2.87E+0	**1.31E–2**	3.00E+0†	2.61E–2
Stewart et al. (2009)	1.54E+1	5.05E–1	1.18E+3	**6.13E–3**	1.58E+3	4.18E–2	1.49E+3	8.59E–2
Ten Tusscher et al. (2004) – Endo	1.78E–3	2.10E+0	1.00E+0†	**3.40E–3**	1.00E+0†	3.00E–2	1.00E+0†	5.81E–2
Ten Tusscher et al. (2004) – Epi	1.78E–3	2.14E+0	1.00E+0†	**3.42E–3**	1.00E+0†	2.98E–2	1.00E+0†	5.87E–2
Ten Tusscher et al. (2004) – M-cell	1.76E–3	1.58E+0	1.00E+0†	**2.54E–3**	1.00E+0†	2.26E–2	1.00E+0†	4.35E–2
Ten Tusscher et al. (2006) – Endo	1.62E–3	1.54E+0	9.45E–1	**2.42E–3**	1.00E+0†	2.13E–2	1.00E+0†	4.29E–2
Ten Tusscher et al. (2006) – Epi	2.14E–3	1.17E+0	1.00E+0†	**2.31E–3**	1.00E+0†	2.13E–2	1.00E+0†	4.38E–2
Ten Tusscher et al. (2006) – M-cell	2.06E–3	1.22E+0	1.00E+0†	**2.25E–3**	1.00E+0†	2.16E–2	1.00E+0†	4.20E–2
Wang–Sobie (2008)	1.66E–2	6.91E–2	5.27E–2	**1.89E–2**	4.46E–1	2.51E–2	6.14E–1	3.63E–2
Winslow31	1.07E–4	1.65E+1	1.07E–4	1.81E+1	9.93E–4	2.01E+1	5.27E–3	**7.68E+0**
Zhang et al. (2000)	9.97E–2	5.78E–2	3.77E+1	2.13E–4	1.00E+3	**1.52E–4**	1.00E+3	1.85E–4

Table A.2: Stepsize, in milliseconds, and execution time, in seconds, of the Backward Euler method using the largest stepsize that produced less than 5% RRMS error.

Model	BE	
	Δt	Time
Courtemanche et al. (1998)	2.30E–1	4.58E+0
Faber–Rudy (2000)	2.91E–1	1.85E+0
Noble (1962)	5.75E–1	3.43E–1
Noble et al. (1998)	3.33E–1	2.53E+0
Pandit et al. (2003)	9.77E–2	4.16E+0
Winslow31	2.24E–2	2.88E+1

Table A.3: Stiffness intervals and execution time, in seconds, of type-insensitive methods using the largest stepsize that produced less than 5% RRMS error. The shortest execution time has been highlighted in bold text for each model.

Model	Stiff Interval	Non-stiff Interval	RL-FE	GRL1-FE	GRL2-FE	BE-FE
			Time			
Bondarenko et al. (2004)	[20 30]	[0 20]; [30 75]	7.10E–1	5.04E–1	**6.86E–2**	1.01E+0
Jafri et al. (1998)	[0 50]	[50 300]	1.06E+0	**6.94E–1**	1.04E+0	2.67E+0
Pandit et al. (2001)	[105 125]	[0 105]; [125 250]	9.16E+0	1.05E–1	**9.94E–2**	2.88E–1
Winslow31	[0 50]	[50 300]	3.02E+0	3.60E+0	**1.54E+0**	4.38E+0

Table A.4: Stepsize, in milliseconds, and execution time, in seconds, of the four numerical methods using the largest stepsize that produced less than 1% RRMS error. The shortest execution time has been highlighted in bold text for each model. Stepsizes that have reached the maximum stepsize are marked by a dagger.

Model	FE		RL		GRL1		GRL2	
	Δt	Time	Δt	Time	Δt	Time	Δt	Time
Beeler–Reuler (1977)	2.53E–2	4.34E–2	1.00E+0†	**1.02E–3**	1.00E+0†	3.08E–3	1.00E+0†	6.00E–3
Bondarenko et al. (2004)	2.13E–4	2.65E+0	2.13E–4	2.30E+0	7.48E–3	8.50E–1	2.85E–2	**4.44E–1**
Courtemanche et al. (1998)	1.94E–2	2.37E–1	5.91E–1	**7.43E–3**	2.00E+0†	1.49E–2	1.94E+0	2.67E–2
Demir et al. (1994)	5.95E–2	1.84E–2	1.53E–1	**6.20E–3**	7.00E–1	1.53E–2	1.99E+0	1.14E–2
Demir et al. (1999)	5.98E–2	1.95E–2	1.53E–1	**8.55E–3**	7.04E–1	1.77E–2	2.00E+0	1.35E–2
DiFrancesco–Noble (1985)	7.92E–2	9.64E–2	3.57E+0	**1.95E–3**	3.21E+0	2.10E–2	7.47E+0	1.89E–2
Dokos et al. (1996)	7.02E–2	3.28E–2	1.27E+0	**1.65E–3**	8.30E–1	2.69E–2	2.95E+0	1.79E–2
Faber–Rudy (2000)	1.12E–2	2.38E–1	5.00E–1†	**4.77E–3**	4.99E–1	5.20E–2	5.00E–1†	1.08E–1
FitzHugh–Nagumo (1961)	2.48E–1	**7.33E–4**	NA	NA	2.50E–2	1.53E–2	2.45E–1	2.74E–3
Fox et al. (2002)	4.62E–3	3.56E–1	5.00E–1	**2.92E–3**	1.00E+0†	1.01E–2	1.00E+0†	2.11E–2
Hilgemann–Noble (1987)	6.25E–2	2.45E–2	8.06E–2	1.51E–2	2.24E+0	6.79E–3	6.77E+0	**4.80E–3**
Hund–Rudy (2004)	1.11E–2	2.51E–1	7.60E–2	**3.43E–2**	8.90E–2	3.03E–1	2.49E–1	2.15E–1
Jafri et al. (1998)	5.76E–4	4.15E+0	5.33E–4	3.88E+0	1.03E–2	**2.31E+0**	1.03E–2	4.75E+0
Luo–Rudy (1991)	1.35E–2	1.51E–1	3.07E–1	5.41E–3	1.00E+0†	**3.19E–3**	1.00E+0†	1.33E–2
Maleckar et al. (2008)	5.02E–2	9.33E–2	8.87E–2	4.61E–2	1.99E+0	2.79E–2	6.00E+0†	**1.89E–2**
McAllister et al. (1975)	2.48E–2	9.27E–2	1.30E+0	**1.63E–3**	1.96E+0	3.15E–3	9.29E–1	2.53E–2
Noble (1962)	2.04E–1	4.13E–3	3.09E–1	**1.79E–3**	2.48E–1	7.88E–3	6.09E–1	6.11E–3
Noble–Noble (1984)	2.04E–1	6.66E–3	2.01E+0	**6.59E–4**	1.41E+0	8.39E–3	5.11E+0	4.81E–3
Noble et al. (1991)	5.15E–2	2.60E–2	1.53E–1	**7.45E–3**	6.56E–1	1.91E–2	1.54E+0	1.61E–2
Noble et al. (1998)	5.58E–2	6.10E–2	1.57E–1	**1.96E–2**	4.72E–1	6.61E–2	1.38E+0	4.71E–2
Nygren et al. (1998)	5.36E–2	1.12E–1	8.88E–2	**5.94E–2**	5.97E–1	9.71E–2	1.60E+0	7.69E–2
Pandit et al. (2001)	2.91E–4	5.95E+0	2.91E–4	5.14E+0	1.08E–1	**1.34E–1**	9.58E–2	3.01E–1
Pandit et al. (2003)	2.65E–5	6.34E+1	2.65E–5	5.68E+1	2.05E–1	**7.48E–2**	1.96E–1	1.51E–1
Puglisi–Bers (2001)	1.08E–2	9.84E–1	2.02E–1	**5.36E–2**	6.14E–1	5.45E–2	6.24E–1	1.11E–1
Sakmann et al. (2000) – Endo	6.90E–2	5.80E–2	1.35E–1	**2.57E–2**	1.19E–1	3.09E–1	3.37E–1	2.25E–1
Sakmann et al. (2000) – Epi	6.90E–2	5.87E–2	1.50E–1	**2.32E–2**	1.56E–1	2.35E–1	4.28E–1	1.79E–1
Sakmann et al. (2000) – M-cell	6.86E–2	5.90E–2	2.36E–1	**1.48E–2**	1.55E+0	2.38E–2	2.80E+0	2.77E–2
Stewart et al. (2009)	1.54E+1	5.09E–1	2.38E+2	**3.00E–2**	2.09E+2	3.11E–1	5.40E+2	2.40E–1
Ten Tusscher et al. (2004) –Endo	1.78E–3	2.11E+0	4.56E–1	**7.30E–3**	1.00E+0†	3.01E–2	1.00E+0†	5.82E–2
Ten Tusscher et al. (2006) –Endo	1.62E–3	1.53E+0	2.58E–1	**8.52E–3**	8.86E–1	2.42E–2	5.56E–1	7.65E–2
Ten Tusscher et al. (2004) –Epi	1.78E–3	2.13E+0	4.55E–1	**7.42E–3**	1.00E+0†	3.03E–2	1.00E+0†	5.97E–2
Ten Tusscher et al. (2006) –Epi	2.14E–3	1.21E+0	3.03E–1	**7.38E–3**	1.00E+0†	2.14E–2	7.89E–1	5.34E–2
Ten Tusscher et al. (2004) –M-cell	1.76E–3	1.60E+0	4.68E–1	**5.35E–3**	8.39E–1	2.69E–2	1.00E+0†	4.47E–2
Ten Tusscher et al. (2006) –M-cell	2.06E–3	1.22E+0	2.73E–1	**8.09E–3**	5.21E–1	4.15E–2	6.10E–1	7.12E–2
Wang–Sobie (2008)	1.66E–2	7.00E–2	5.27E–2	**1.89E–2**	4.46E–1	2.54E–2	5.73E–1	3.89E–2
Winslow31	1.07E–4	**1.65E+1**	1.07E–4	1.81E+1	4.62E–4	4.35E+1	1.73E–3	2.43E+1
Zhang et al. (2000)	9.97E–2	5.85E–2	2.90E+0	**1.91E–3**	2.09E+0	1.74E–2	4.81E+0	1.47E–2

Table A.5: Stepsize, in milliseconds, and execution time, in seconds, of the Backward Euler method using the largest stepsize that produced less than 1% RRMS error.

Model	BE	
	Δt	Time
Courtemanche et al. (1998)	2.30E–1	4.58E+0
Faber–Rudy (2000)	2.91E–1	1.85E+0
Noble (1962)	3.19E–1	6.14E–1
Noble et al. (1998)	3.33E–1	2.53E+0
Pandit et al. (2003)	9.77E–2	4.16E+0
Winslow31	1.76E–2	3.68E+1

Table A.6: Stiffness intervals and execution time, in seconds, of type-insensitive methods using the largest stepsize that produced less than 1% RRMS error. The shortest execution time has been highlighted in bold text for each model.

Model	Stiff Interval	Non-stiff Interval	RL-FE	GRL1-FE	GRL2-FE	BE-FE
			Time			
Bondarenko et al. (2004)	[20 30]	[0 20]; [30 75]	7.10E–1	5.08E–1	**6.85E–2**	1.01E+0
Jafri et al. (1998)	[0 50]	[50 300]	1.05E+0	**7.54E–1**	1.05E+0	2.67E+0
Pandit et al. (2001)	[105 125]	[0 105]; [125 250]	9.28E+0	1.04E–1	**1.01E–1**	2.88E–1
Winslow31	[0 50]	[50 300]	3.10E+0	7.79E+0	**1.65E+0**	5.38E+0

Table A.7: Stepsize, in milliseconds, and execution time, in seconds, of the four numerical methods using the largest stepsize that produced less than 1% MRMS error. The shortest execution time has been highlighted in bold text for each model.

Model	FE		RL		GRL1		GRL2	
	Δt	Time	Δt	Time	Δt	Time	Δt	Time
Beeler–Reuler (1977)	2.53E–2	4.23E–2	6.81E–2	1.41E–2	1.36E–1	2.20E–2	5.10E–1	**1.14E–2**
Bondarenko et al. (2004)	2.13E–4	2.65E+0	2.13E–4	**2.30E+0**	2.53E–3	2.51E+0	1.34E–3	9.51E+0
Courtemanche et al. (1998)	1.94E–2	**2.35E–1**	1.77E–2	2.47E–1	2.01E–2	1.43E+0	1.61E–1	3.27E–1
Demir et al. (1994)	2.90E–2	**3.75E–2**	1.09E–2	8.51E–2	2.34E–2	4.53E–1	1.49E–1	1.50E–1
Demir et al. (1999)	2.48E–2	**4.65E–2**	9.73E–3	1.32E–1	1.99E–2	6.27E–1	1.50E–1	1.78E–1
DiFrancesco–Noble (1985)	7.65E–2	**1.00E–1**	4.38E–2	1.51E–1	4.63E–2	1.43E+0	2.78E–1	5.03E–1
Dokos et al. (1996)	6.91E–2	**3.39E–2**	2.66E–2	7.50E–2	1.71E–2	1.29E+0	2.88E–1	1.80E–1
FitzHugh–Nagumo (1961)	5.41E–4	**3.18E–1**	NA	NA	5.18E–4	7.12E–1	5.20E–4	1.15E+0
Faber–Rudy (2000)	6.60E–3	**4.13E–1**	3.79E–3	6.27E–1	6.99E–3	3.69E+0	1.14E–1	4.74E–1
Fox et al. (2002)	4.62E–3	3.57E–1	9.08E–3	**1.57E–1**	2.22E–2	4.42E–1	2.20E–2	9.51E–1
Hilgemann–Noble (1987)	6.25E–2	**2.49E–2**	4.71E–2	2.59E–2	2.69E–2	5.49E–1	2.07E–1	1.52E–1
Hund–Rudy (2004)	1.58E–3	**1.77E+0**	1.06E–3	2.45E+00	1.09E–3	2.49E+1	2.94E–2	1.85E+0
Jafri et al. (1998)	5.27E–4	4.55E+0	5.27E–4	**3.93E+0**	3.75E–4	6.38E+1	1.57E–3	3.09E+1
Luo–Rudy (1991)	1.35E–2	1.48E–1	3.82E–2	**4.31E–2**	3.37E–2	9.45E–2	1.86E–1	7.00E–2
Maleckar et al. (2008)	5.01E–2	9.40E–2	8.87E–2	**4.59E–2**	9.02E–2	6.03E–1	2.30E–1	4.86E–1
McAllister et al. (1975)	1.10E–2	2.10E–1	5.62E–2	**3.61E–2**	4.27E–2	1.39E–1	6.13E–2	3.64E–1
Noble (1962)	7.64E–2	**1.08E–2**	2.25E–2	2.35E–2	1.62E–2	1.18E–1	9.61E–2	3.85E–2
Noble–Noble (1984)	5.48E–2	**2.46E–2**	2.43E–2	4.70E–2	1.84E–2	6.15E–1	4.01E–1	5.94E–2
Noble et al. (1991)	5.15E–2	2.57E–2	6.79E–2	**1.67E–2**	1.89E–2	6.54E–1	1.37E–1	1.78E–1
Noble et al. (1998)	5.56E–2	6.12E–2	7.83E–2	**3.91E–2**	1.75E–2	1.77E+0	1.30E–1	4.93E–1
Nygren et al. (1998)	5.32E–2	1.13E–1	5.83E–2	**8.95E–2**	4.09E–2	1.42E+0	1.49E–1	8.24E–1
Pandit et al. (2001)	2.91E–4	6.04E+0	2.91E–4	5.14E+0	2.61E–3	5.57E+0	1.10E–2	**2.63E+0**
Pandit et al. (2003)	2.65E–5	6.34E+1	2.65E–5	5.68E+1	1.41E–3	1.09E+1	3.09E–3	**9.58E+0**
Puglisi–Bers (2001)	3.68E–3	3.05E+0	2.84E–3	4.12E+0	6.30E–3	5.41E+0	2.14E–1	**3.34E–1**
Sakmann et al. (2000) – Endo	3.14E–2	**1.29E–1**	1.58E–2	2.18E–1	1.13E–2	3.24E+0	7.99E–2	9.58E–1
Sakmann et al. (2000) – Epi	2.66E–2	**1.52E–1**	1.16E–2	2.98E–1	1.00E–2	3.67E+0	7.44E–2	1.04E+0
Sakmann et al. (2000) – M-cell	6.86E–2	5.87E–2	9.21E–2	**3.77E–2**	8.50E–2	4.45E–1	3.31E–1	2.31E–1
Stewart et al. (2009)	1.46E+1	5.41E–1	4.66E+1	**1.52E–1**	3.67E+1	1.77E+0	1.28E+2	9.97E–1
Ten Tusscher et al. (2004) –Endo	1.78E–3	2.13E+0	2.89E–2	**1.13E–1**	2.87E–2	1.05E+0	1.63E–1	3.63E–1
Ten Tusscher et al. (2006) –Endo	1.62E–3	1.54E+0	1.69E–2	**1.28E–1**	3.20E–2	6.74E–1	8.87E–2	4.81E–1
Ten Tusscher et al. (2004) –Epi	1.78E–3	2.15E+0	2.76E–2	**1.20E–1**	2.58E–2	1.16E+0	1.64E–1	3.58E–1
Ten Tusscher et al. (2006) –Epi	2.14E–3	1.19E+0	3.22E–2	**6.81E–2**	3.83E–2	5.62E–1	1.63E–1	2.63E–1
Ten Tusscher et al. (2004) –M-cell	1.76E–3	1.59E+0	2.75E–2	**8.87E–2**	2.13E–2	1.04E+0	2.45E–1	1.79E–1
Ten Tusscher et al. (2006) –M-cell	2.06E–3	1.23E+0	3.03E–2	**7.16E–2**	2.29E–2	9.32E–1	1.41E–1	3.01E–1
Wang–Sobie (2008)	1.66E–2	6.91E–2	1.63E–2	**6.05E–2**	1.74E–2	6.48E–1	1.59E–1	1.40E–1
Winslow31	1.07E–4	**1.65E+1**	1.07E–4	1.81E+1	1.95E–5	1.06E+3	3.07E-4	1.39E+2
Zhang et al. (2000)	9.97E–2	5.86E–2	1.05E–1	**5.01E–2**	6.74E–2	4.98E–1	5.66E–1	1.21E–1

Table A.8: Stepsize, in milliseconds, and execution time, in seconds, of the Backward Euler method using the largest stepsize that produced less than 1% MRMS error.

Model	BE	
	Δt	Time
Courtemanche et al. (1998)	3.19E–2	2.55E+1
Faber–Rudy (2000)	6.51E–3	9.41E+1
Noble (1962)	6.48E–2	2.35E+0
Noble et al. (1998)	8.20E–2	8.70E+0
Pandit et al. (2003)	1.24E–2	2.46E+1
Winslow31	7.32E–4	5.18E+2

Table A.9: Stiffness intervals and execution time, in seconds, of type-insensitive methods using the largest stepsize that produced less than 1% MRMS error. The shortest execution time has been highlighted in bold text for each model.

Model	Stiff Interval	Non-stiff Interval	RL-FE	GRL1-FE	GRL2-FE	BE-FE
			Time			
Bondarenko et al. (2004)	[20 30]	[0 20]; [30 75]	7.27E–1	**7.26E–1**	1.67E+0	9.12E+0
Jafri et al. (1998)	[0 50]	[50 300]	**1.16E+0**	1.25E+1	4.88E+0	2.26E+1
Pandit et al. (2001)	[105 125]	[0 105]; [125 250]	9.66E+0	1.21E–1	**1.03E–1**	3.46E+0
Winslow31	[0 50]	[50 300]	**3.11E+0**	1.91E+2	2.95E+1	9.10E+1