Numerical methods for simulation of electrical activity in the myocardial tissue

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 Computer Science
University of Saskatchewan
Saskatoon

By
Ryan C. Dean

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 Computer Science
176 Thorvaldson Building
110 Science Place
University of Saskatchewan
Saskatoon, Saskatchewan
Canada
S7N 5C9

Abstract

Mathematical models of electric activity in cardiac tissue are becoming increasingly powerful tools in the study of cardiac arrhythmias. Considered here are mathematical models based on ordinary differential equations (ODEs) and partial differential equations (PDEs) that describe the behaviour of this electrical activity. Generating an efficient numerical solution of these models is a challenging task, and in fact the physiological accuracy of tissue-scale models is often limited by the efficiency of the numerical solution process. In this thesis, we discuss two set of experiments that test ideas for making the numerical solution process more efficient. In the first set of experiments, we examine the numerical solution of four single cell cardiac electrophysiological models, which consist solely of ODEs. We study the efficiency of using implicit-explicit Runge–Kutta (IMEX-RK) splitting methods to solve these models. We find that variable step-size implementations of IMEX-RK methods (ARK3 and ARK5) that take advantage of Jacobian structure clearly outperform most methods commonly used in practice for two of the models, and they outperform all methods commonly used in practice for the remaining models. In the second set of experiments, we examine the solution of the bidomain model, a model consisting of both ODEs and PDEs that are typically solved separately. We focus these experiments on numerical methods for the solution of the two PDEs in the bidomain model. The most popular method for this task, Crank–Nicolson, produces unphysical oscillations; we propose a method based on a second-order L-stable singly diagonally implicit Runge–Kutta (SDIRK) method to eliminate these oscillations. We find that although the SDIRK method is able to eliminate these unphysical oscillations, it is only more efficient for crude error tolerances.

Acknowledgements

Acknowledgements go here. Typically you would at least thank your supervisor.

Contents

List of Tables

List of Figures

List of Abbreviations

BDF	Backward Differentiation Formula
BE	Backward Euler
CN	Crank–Nicolson
CRT	Model of Courtemanche et al.
DIRK	Diagonally Implicit Runge–Kutta
DP	Dormand Prince
ERK	Explicit Runge–Kutta
ESDIRK	Explicit Singly Diagonally Implicit Runge–Kutta
FE	Forward Euler
FEM	Finite Element Method
IMEX	Implicit-Explicit
IMEX-RK	Implicit-Explicit Runge–Kutta
IRK	Implicit Runge–Kutta
IVP	Initial Value Problem
LMM	Linear Multistep Method
LR	Luo–Rudy model
NSFD	Non-Standard Finite Difference
ODE	Ordinary Differential Equation
PB	Puglisi–Bers model
PDE	Partial Differential Equation
RK	Runge–Kutta
RL	Rush–Larsen
RRMS	Relative Root Mean Square
SDIRK	Singly Diagonally Implicit Runge–Kutta
WIN	Model of Winslow et al.
WR	Waveform Relaxation

Chapter 1

Introduction

Computer simulation is becoming an important tool in cardiovascular research. Mathematical models of the heart can be used to simulate heart conditions and the effects of certain drugs designed to treat them. Presently, the development of a drug often costs hundreds of millions of dollars [14]. One aim of computer simulation is to reduce this cost, e.g., by reducing the number of physical experiments needed in designing a drug.

Electrophysiological models of the heart describe how electricity flows through the heart, controlling its contraction. The models in which we are interested consist of systems of differential equations. Models of the electrophysiology of one cell are governed by systems of ordinary differential equations (ODEs), and models of the electrophysiology of more than one cell are governed by one or more partial differential equations (PDEs). Typically, a PDE model is coupled with an ODE model to simulate heart tissue consisting of a network of cells; the ODEs model the electrical activity in the cells, and the PDEs model propagation of the electrical activity across the network as a whole. Cardiac electrophysiological models are often based on the Nobel prize-winning work of Hodgkin and Huxley [23] in the 1950s that modelled neural tissue mathematically as a circuit. Modern cardiac electrophysiological models adapt the work of Hodgkin and Huxley to describe electrical activity in the heart and include data gathered from experiments to form models with increasing physiological accuracy.

A major barrier to obtaining the most useful data from tissue-scale electrophysiological models of the heart is the challenge of performing the simulations efficiently. Often the physiological accuracy of the mathematical model must be sacrificed for a simulation to become feasible; see, e.g., [53, 65, 67, 60]. The ODE systems describing the cellular dynamics in single cell models are non-linear and stiff; see, e.g., [27]. The consequence of stiffness is that the speed of the solution process is limited by considerations of numerical stability instead of accuracy. Hence, the solution process can potentially be made more efficient through the use of appropriate numerical algorithms. To this end, we present two sets of numerical experiments. Each set of experiments tests an idea for making the solution process more efficient.

In the first set of experiments, we examine the use of implicit-explicit Runge–Kutta (IMEX-RK) methods [4] for solving single cell cardiac electrophysiological models. An IMEX-RK method

approximates the solution of a differential equation which is split into two parts: one part better suited for solution by an implicit method and the other by an explicit method. An IMEX-RK method uses both an implicit and an explicit method to approximate the solution to the respective parts. Using these methods together, we are able to maximize the efficiency of the solution by using the method best suited to each part. We examine the numerical solution of four ODE models and compare the efficiency of two particular IMEX-RK methods to ten numerical methods commonly used in practice.

In the second set of experiments, we examine one step in the solution process of the bidomain model. The bidomain model is a system of coupled ODEs and PDEs that are typically solved separately using a process called operator splitting. These experiments focus on the solution of the two PDEs in the bidomain model. Second-order methods can generally be expected to be more effective than first-order methods; however, the use of the Crank–Nicolson (CN) method, one of the most common second-order methods for this task, can lead to solutions with noticeable unphysical oscillations. As an alternative, we propose the use of a second-order L-stable singly diagonally implicit Runge–Kutta method (SDIRK) to eliminate the unphysical oscillations. We examine the performance of the SDIRK method in a scenario in which CN is known to produce unphysical oscillations.

The rest of this thesis is organized as follows. Chapter 2 gives an introduction to cardiac electrophysiology and discusses five popular ODE cell models. Chapter 3 gives an introduction to the numerical solution of differential equations, both ODEs and PDEs, and a review of numerical methods used to simulate cardiac electrophysiological models. Chapter 4 outlines numerical experiments and gives the results. In the first set of experiments, IMEX-RK methods are found to outperform all methods of which we are aware for two of the four models studied. In the second set of experiments, the SDIRK method is found to eliminate unphysical oscillations but is more efficient only for crude error tolerances. Finally, Chapter 5 gives some conclusions and suggestions for future work.

Chapter 2

Electrical Activity in the Heart

2.1 Physiological Background

Electrical activity is responsible for the periodic contraction and relaxation cycle of the heart that propels blood throughout the body [56]. Hence, electrical activity is essential for the heart to perform its function. Most serious heart problems are in fact related to disturbances in the heart's electrical activity [56].

The heart has approximately 10^{10} cells, and every one of them has an electrical potential [56]. Due to the different net electrical charges of different ions in the cytoplasm of heart cells, a heart cell is negatively charged with respect to its surroundings. This results in a potential difference across the cell membrane, called the *transmembrane potential*. If an electrical stimulus applied to the cell is able to raise the transmembrane potential above a certain threshold, then the cell's conduction properties change to allow positive ions to flow into the cell, causing a reversal in the potential difference across the cell membrane. This results in the *depolarization* of the cell, during which time the transmembrane potential increases significantly. After depolarization, there is a *plateau phase*, during which the cell remains depolarized. It is during this phase that the blood is pumped [48]. The cell then enters a *repolarization phase*, during which the cell's potential returns to its resting potential. The natural cycle from depolarization to repolarization, called an *action potential*, begins with a spontaneous electrical pulse emanating from specialized tissue called the *sinoatrial node*.

Many heart problems are the result of irregularities in the flow of electricity in the heart. Abnormal electrical activity is called an *arrhythmia*, which in general is caused either by abnormal impulse formation, abnormal conduction, or *re-entry* [2]. Re-entry is the result of an electrical impulse that persists past the normal activation of the heart and re-excites tissue that has already contracted during the current heartbeat. This causes irregularities in the heartbeat that can lead to a number of serious problems, including death. To be able to study these conditions noninvasively is one common practical motivation for creating mathematical models of electrical activity in cardiac tissue.

The specific objective of these mathematical models is to model the heart and heart conditions

in order to simulate treatments; e.g., we can use these mathematical models to perform computer simulations of the effects of new drugs to treat these heart conditions. Presently, the development of a new drug has an average cost of approximately $900 million [14]; one objective of using computer simulation to simulate new drugs is to reduce the number of physical experiments needed to develop the drug and therefore reduce this cost. As the models become more physiologically accurate we are able to obtain more useful information from these simulations.

Because of their intricacy, obtaining physiologically accurate mathematical models is a difficult task. A further challenge to obtaining physiological accuracy is that of performing the simulation efficiently. To move effectively beyond models for one cell, enough cells must be included in the model to realistically approximate the geometry and physiology of the heart. Because the heart has approximately 10^{10} cells [56], any realistic simulation will have enough cells (or clusters of cells) to dramatically magnify any inefficiencies in the numerical method. This has forced some researchers to reduce the physiological accuracy of their models to allow the simulation to be performed within an acceptable amount of time; see e.g., [53, 65, 67, 60]. The models are numerically *stiff*, and so standard (explicit) numerical methods are often unable to provide efficient simulations [20]. If the efficiency of the simulation process can be significantly improved, then greater physiological accuracy and subsequently more useful data can be obtained.

2.2 Single Cell Models

The behaviour of electrical activity can be modelled with something as simple as a cubic polynomial [56]. For example, the function

$$f(V_m) = A^2(V_m - V_{rest})(V_m - V_{th})(V_m - V_{peak}), \tag{2.2.1}$$

can reproduce macroscopically observed behaviour of electrical activity in the cell [56]. Here V_m is the transmembrane potential, A is the rate of change of the transmembrane potential during the depolarization phase, V_{rest} is the resting potential, V_{th} is the threshold potential, and V_{peak} is the maximum potential. All potentials are measured in mV, and A is measured in mS mm^{-1}. Models of this type are called *phenomenological models* [56]. Phenomenological models reproduce only the macroscopic details regarding electrical activity and do not include any of the underlying physiological details that cause the creation and behaviour of electrical activity in the heart. For example, equation (2.2.1) does not model the re-polarization phase [56]. The advantage to using phenomenological models is that they can simulate an action potential with the lowest possible computational cost [46].

More typically, ordinary differential equations (ODEs) are used to model the behaviour of

electricity in a myocardial cell[1]. ODE models can be placed into two groups.

First-generation models include behaviour described by a phenomenological model as well as some of the underlying physiology [56]. The ionic channels most responsible for generating an action potential are included in a first-generation model, but many of the finer details are simplified. Examples of first-generation models are the FitzHugh–Nagumo and Luo–Rudy Phase I models, discussed in sections 2.2.1 and 2.2.2.

Second-generation models include all of the detail of a first-generation model and as many of the finer details as possible [56]. Although models as simple as equation (2.2.1) can be useful in certain cases, the most useful simulations require second-generation models because details on the finest level can affect the behaviour of the heart as a whole [56]. Examples of second-generation models are the models of Courtemanche et al., Winslow et al., and Puglisi–Bers discussed in sections 2.2.3, 2.2.4, and 2.2.5, respectively.

Another way of classifying cardiac electrophysiological models is based on what is being modelled [46]. For example, we have sinoatrial node models, atrial models, atrioventricular models, Purkinje fibre models, and ventricular models. See, e.g., [46] for a detailed list of dozens of cardiac electrophysiological models classified in this way.

2.2.1 The FitzHugh–Nagumo model

One of the simplest single cell models is what is now called the FitzHugh–Nagumo (FHN) model. The model was originally developed as simplification of the Hodgkin–Huxley model by FitzHugh in 1961 [17] and expressed in the equivalent circuit by Nagumo in 1962 [40]. One could also view the FHN model as adding just enough detail to phenomenological models, such as equation (2.2.1), to overcome the failure to model the repolarization phase. The original two-variable formulation is given by

$$\frac{dV_m}{dt} = c_1 V_m (V_m - a)(1 - V_m) - c_2 w + I_{st}, \tag{2.2.2}$$

$$\frac{dw}{dt} = b(V_m - c_3 w), \tag{2.2.3}$$

where V_m is the transmembrane potential, measured in mV, w is a dimensionless recovery variable, I_{st} is the stimulus current and a, b, c_1, c_2, and c_3 are parameters to the model. These parameters may be modified to model different cell types [56]. An example of values for these parameters is given in Table 2.2.1.

The FHN model is simple to implement and computationally inexpensive but it is limited in terms of the physiological accuracy. For example, in the FHN model the cell hyperpolarizes during

[1]The reader unfamiliar with differential equations should read section 3.1 first.

the repolarization phase, something that does not occur in physiological data [56]. Many attempts have been made to modify the FHN model to make it more physiologically accurate while retaining the simplicity of the original model. One such modification by Rogers and McCulloch [49] changes one term in equation (2.2.2) to eliminate the hyperpolarization:

$$\frac{dV_m}{dt} = c_1 V_m(V_m - a)(1 - V_m) - c_2 V_m w + I_{st}, \tag{2.2.4}$$

$$\frac{dw}{dt} = b(V_m - c_3 w). \tag{2.2.5}$$

Examples of other modifications include [1, 32, 8].

The FHN model normalizes values of the transmembrane potential to be in the range $[-1, 1]$ (or in the range $[0, 1]$ for the modified model); a change of variables is needed so that the transmembrane potential has realistic values [46, 56]. First, we define the total amplitude, V_{amp}, in terms of the peak and resting potentials: $V_{amp} = V_{peak} - V_{rest}$. This gives the definition of two new variables [56]

$$V = V_{amp} V_m + V_{rest}, \tag{2.2.6}$$

$$W = v_{amp} w, \tag{2.2.7}$$

that, when substituted into equations (2.2.4)–(2.2.5), transforms the FHN model into a form that produces realistic transmembrane potentials [56]:

$$\frac{dV}{dt} = \frac{c_1}{V_{amp}^2}(V - V_{rest})(V - V_{th})(V_{peak} - V) - \frac{c_2}{V_{amp}}(V - V_{rest})W + I_{st}, \tag{2.2.8}$$

$$\frac{dW}{dt} = b(V - V_{rest} - c_3 w), \tag{2.2.9}$$

where the threshold potential is defined by $V_{th} = V_{rest} + aV_{amp}$. Further references to the FHN model in this thesis refer to the formulation in equations (2.2.8)-(2.2.9).

Table 2.1: An example of parameters to the FHN model [46, 56]

Parameter	Value	Units
a	0.13	dimensionless
b	0.013	dimensionless
c_1	0.26	ms^{-1}
c_2	0.1	ms^{-1}
c_3	1.0	ms^{-1}
V_{rest}	–85	mV
V_{peak}	40	mV

2.2.2 The Luo–Rudy model

In 1991 Luo and Rudy developed a model of guinea pig ventricular action potentials based on a previous model from Beeler and Reuter [5]. The Luo–Rudy model [34] extended the Beeler–Reuter model to include fast inward sodium and outward potassium currents to make the model more physiologically accurate. The general approach of these models is based on Hodgkin–Huxley type formalism [23]; the Luo–Rudy model itself consists of 8 nonlinear ODEs.

The quantity of primary concern is the transmembrane potential, due to its importance as discussed in section 2.1. For an individual cardiac cell we have that the transmembrane potential V_m, typically measured in mV, satisfies [34]:

$$\frac{dV_m}{dt} = -\frac{1}{C_m}(I_{ion} + I_{st}), \tag{2.2.10}$$

where C_m is the membrane capacitance, I_{ion} is the total transmembrane ionic current, and I_{st} is the stimulus current.

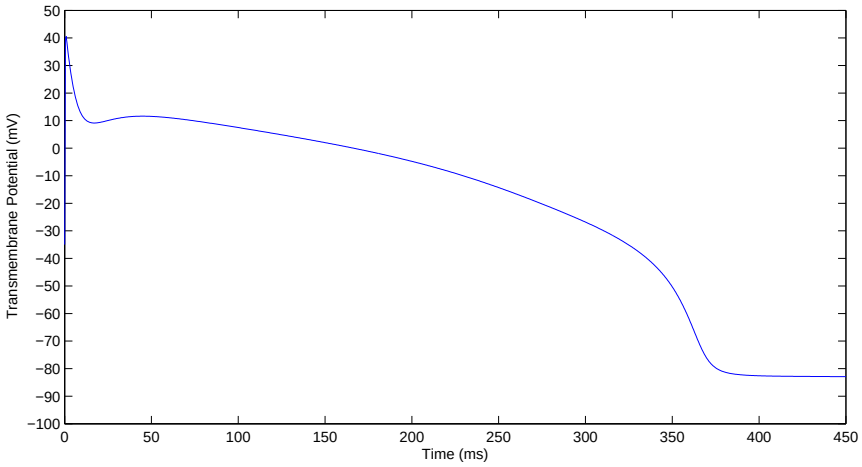


Figure 2.1: Transmembrane potential over time in the Luo–Rudy model. This is a numerical solution for equation (2.2.10) coupled with 7 other ODEs [34] via the I_{ion} current.

An example of the evolution of V_m over time for this model is given in Figure 2.1. Note that the sign of I_{ion} determines the direction in which positive ions are flowing [30]. For example, a negative value for I_{ion} means that we have a net inward flow of positive ions, and so the potential has a positive derivative, and the potential is increasing [30]. A positive value for I_{ion} means that there is a net outward flow of positive ions, and so the potential has a negative derivative, and the potential is decreasing.

The Luo–Rudy model contains 6 ionic currents that are determined by 6 gating variables [34].

The evolution of each dimensionless gating variable y is governed by a nonlinear ODE involving rate parameters α_y and β_y in the general form

$$\frac{dy}{dt} = \frac{y_\infty - y}{\tau_y}, \tag{2.2.11}$$

where

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

are the steady state and time constant values of y, respectively. Equivalently, gating equations are sometimes written in the form

$$\frac{dy}{dt} = \alpha_y(1 - y) - \beta_y y. \tag{2.2.12}$$

The remaining ODE in the Luo–Rudy model describes calcium concentration in the cell [34]:

$$\frac{d\left([\mathrm{Ca}]_i\right)}{dt} = -10^{-4} I_{\mathrm{si}} + 0.07(10^{-4} - [\mathrm{Ca}]_i), \tag{2.2.13}$$

where $[\mathrm{Ca}]_i$ is the intracellular calcium concentration, measured in mM, and I_{si} is the slow inward calcium current, measured in $\mu\mathrm{A/cm}^2$. The 6 gating equations of the form (2.2.11) are coupled with (2.2.10) and (2.2.13) to form the complete Luo–Rudy model. A complete listing of the governing equations can be found in Appendix A.1. Full details of the model can be found in [34].

In 1994 Luo and Rudy published an improvement to this model, now known as the Luo–Rudy Phase II model [35, 36]. This model includes the actions of ionic pumps and changes in ionic concentrations. Consisting of 14 ODEs, it is a more physiologically accurate yet more complicated model than that given by (2.2.10), (2.2.11), and (2.2.13). We do not consider the Luo–Rudy Phase II model in this study.

2.2.3 The model of Courtemanche et al.

In 1998 Courtemanche, Ramirez, and Nattel developed a model of human atrial action potentials [11]. It was developed in response to findings that show there are important differences in human action potentials when compared to those of other mammals frequently used in models. Courtemanche et al. developed this model with human data supplemented with animal data when needed. The model of Courtemanche et al. is an extension of the Luo–Rudy Phase II model. It consists of 21 ODEs. A complete listing of the governing equations can be found in Appendix A.2. Full details of the model can be found in [11].

2.2.4 The model of Winslow et al.

In 1999 Winslow, Rice, Jafri, Marbán, and O'Rourke developed a model of canine ventricular tissue [69]. This model is based on a guinea pig model that was an extension of the Luo–Rudy Phase II model. The model of Winslow et al. was developed using experimental data to modify the guinea pig model so that it would simulate canine ventricular tissue. The model of Winslow et al. is particularly detailed when describing the dynamics of Ca^{2+}, which is an important consideration in heart failure. It consists of 32 ODEs, making it the most complex of the models in this study. A complete listing of the governing equations can be found in Appendix A.3. Full details of the model can be found in [69].

2.2.5 The Puglisi–Bers model

In 2001 Puglisi and Bers developed a model of rabbit ventricular tissue [45]. Although rabbit ventricular tissue is used frequently in experiment, no mathematical model had been previously developed for it. This model was adapted from the Luo–Rudy model to include data from the literature and from the joint laboratory of Puglisi and Bers. This model was designed to be a learning aid for students as well as a tool for researchers to reproduce experimental data via computer simulation. Thus, physiological accuracy was of paramount importance. The Puglisi–Bers model gives particular detail to calcium handling in order to accurately simulate heart failure. This model contains 17 ODEs. It is also referred to as the *LabHeart* model [45]. A complete listing of the governing equations can be found in Appendix A.4. Full details of the model can be found in [45].

2.3 PDE Models

In order to describe electrical activity in the whole heart, we may couple a single cell model with a PDE model that describes how electricity flows across a network of cells [56]. One PDE model, called the *bidomain model*, is introduced in section 2.3.1. The bidomain model was developed by Leslie Tung in 1978 [64] and is now widely used for simulating electrical activity at the organ level. A common, albeit unrealistic, simplification of the bidomain model, called the *monodomain model*, is discussed in section 2.3.2, and an extension of the bidomain model to include the torso is discussed in section 2.3.3.

2.3.1 Bidomain Model

Heart tissue can be classified into two groups: intracellular and extracellular. To account for the effects of potential differences across the cell membrane, the bidomain model treats these two groups

as two separate domains. Each point in the heart is considered to be in both domains, which can be thought of as superimposed on one another [66]. Each point has an electrical potential defined in each domain [56].

The derivation of the bidomain model begins with Ohm's law. For each domain, Ohm's law requires the current to be

$$\mathbf{J}_I = -\boldsymbol{\sigma}_I \nabla u_I, \tag{2.3.1}$$

$$\mathbf{J}_E = -\boldsymbol{\sigma}_E \nabla u_E, \tag{2.3.2}$$

where $\mathbf{J}_I$ and $\mathbf{J}_E$ are the currents, $\boldsymbol{\sigma}_I$ and $\boldsymbol{\sigma}_E$ are the conductivity tensors, and u_I and u_E are the potentials for the intracellular and extracellular domains, respectively [56].

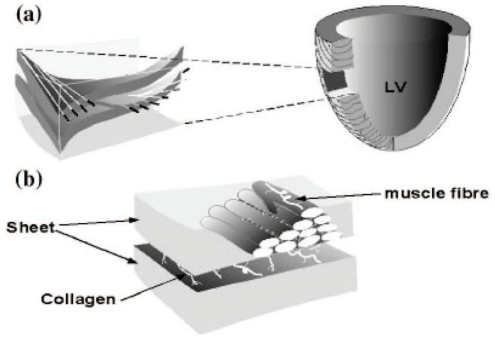


Figure 2.2: Cross section of muscle fibre in the left ventricle (LV), illustrating (a) the change in the direction of muscle fibre throughout the heart and (b) the groupings of fibres and sheets [26].

The conductivity tensors, $\boldsymbol{\sigma}_I$ and $\boldsymbol{\sigma}_E$, are defined largely by the structure of the heart. Cardiac cells are grouped into muscle fibres, and the muscle fibres are grouped into sheets of fibres [56]; see Figure 2.2. These structures influence the flow of electricity; conductivity is greater along the fibres rather than across them [56]. These properties imply three characteristic directions for the conductivity values: parallel to the fibres, perpendicular to the fibres but parallel to the sheet, and perpendicular to the sheet [56]. As the directions of individual fibres change throughout the heart, so do the conductivity values of the heart. Hence, at each point in each domain a local conductivity tensor, denoted $\boldsymbol{\sigma}^*$, can be defined in the basis formed by three perpendicular unit vectors: $\hat{\mathbf{e}}_l, \hat{\mathbf{e}}_\tau$, and $\hat{\mathbf{e}}_n$ for along the fibre, perpendicular to the fibre but parallel to the sheet, and perpendicular to the sheet, respectively. $\boldsymbol{\sigma}^*$ can be expressed as

$$\boldsymbol{\sigma}^* = \begin{bmatrix} \sigma_l & 0 & 0 \\ 0 & \sigma_\tau & 0 \\ 0 & 0 & \sigma_n \end{bmatrix},$$

Table 2.2: The top part of this table describes constants, intermediate values, and variables in the bidomain model. When multiple subscripts are possible, they appear as a set inside braces. The bottom part of this table describes the meaning of the subscripts.

Generic Variable	Units	Description
C_m	μF/cm^2	Capacitance
$\hat{\mathbf{e}}_{\{l,\tau,n\}}$	mS/cm	Unit vector
$\mathbf{J}_{\{I,E\}}$	mA	Current density
$\boldsymbol{\sigma}_{\{I,E\}}$	mS/cm	Conductivity Tensor
$u_{\{I,E\}}$	mV	Potential
V_m	mV	Transmembrane potential
$\sigma_{\{l,\tau,n\}}$	mS/cm	Local conductivity values
χ	cm^{-1}	Membrane area to volume ratio

Subscript	Description
I	Intracellular
E	Extracellular
l	Along fibre
τ	Perpendicular to fibre, parallel to sheet
n	Perpendicular to sheet

where σ_l, σ_τ, and σ_n are local conductivity values expressed in terms of the basis at that point. Given conductivity tensors for each domain expressed in the local coordinate system, denoted $\boldsymbol{\sigma}_I^*$ and $\boldsymbol{\sigma}_E^*$, the global conductivity matrices can be expressed as

$$\boldsymbol{\sigma}_I = \mathbf{E}\boldsymbol{\sigma}_I^*\mathbf{E}^T,$$
$$\boldsymbol{\sigma}_E = \mathbf{E}\boldsymbol{\sigma}_E^*\mathbf{E}^T,$$

where $\mathbf{E}$ is a matrix having $\hat{\mathbf{e}}_l, \hat{\mathbf{e}}_\tau$, and $\hat{\mathbf{e}}_n$ as columns. Hence, the (j,k) entry entry, $\boldsymbol{\sigma}_I(j,k)$, in the global conductivity tensor $\boldsymbol{\sigma}_I$ is given by

$$\boldsymbol{\sigma}_I(j,k) = \hat{e}_l^j \hat{e}_l^k \sigma_l^I + \hat{e}_\tau^j \hat{e}_\tau^k \sigma_\tau^I + \hat{e}_n^j \hat{e}_n^k \sigma_n^I, \tag{2.3.3}$$

for $j,k = 1,2,3$. The global conductivity tensor $\boldsymbol{\sigma}_E$ is defined similarly.

Here we are considering the heart in isolation; any current leaving one domain must enter the other [46]. Hence, a change in current density will be of equal magnitude in both domains but will have the opposite sign:

$$-\nabla \cdot \mathbf{J}_I = \nabla \cdot \mathbf{J}_E. \tag{2.3.4}$$

The current flow across the membrane must be equal to either side of equation (2.3.4). This current flow may be seen as a time-dependent capacitive current together with an ionic current

[46]. Thus, using Kirchoff's current law, we obtain:

$$-\nabla \cdot \mathbf{J}_I = \nabla \cdot \mathbf{J}_E = \chi \left(C_m \frac{\partial V_m}{\partial t} + I_{ion} \right), \tag{2.3.5}$$

where C_m is the capacitance of the cell membrane, I_{ion} is the ionic current across the cell membrane, χ is the area of the cell membrane per unit to volume, and V_m is the transmembrane potential, defined by $V_m = u_I - u_E$.

Using equations (2.3.1) and (2.3.2) with equation (2.3.5) gives

$$\nabla \cdot (\boldsymbol{\sigma}_I \nabla u_I) = \chi \left(C_m \frac{\partial V_m}{\partial t} + I_{ion} \right), \tag{2.3.6}$$

$$\nabla \cdot (\boldsymbol{\sigma}_E \nabla u_E) = -\chi \left(C_m \frac{\partial V_m}{\partial t} + I_{ion} \right). \tag{2.3.7}$$

The difference in signs between (2.3.6) and (2.3.7) is due to defining the positive direction of flow to be from intracellular to extracellular [56]. Thus, adding (2.3.6) and (2.3.7) leads to

$$\nabla \cdot (\boldsymbol{\sigma}_I \nabla u_I) + \nabla \cdot (\boldsymbol{\sigma}_E \nabla u_E) = 0. \tag{2.3.8}$$

Equations (2.3.6) and (2.3.8) describe completely the three potentials of interest: u_E, u_I, and V_m. Using the definition of V_m we can simplify these equations by eliminating the intracellular potential. This gives the standard formulation of the bidomain model [56]:

$$\nabla \cdot (\boldsymbol{\sigma}_I \nabla V_m) + \nabla \cdot (\boldsymbol{\sigma}_I \nabla u_E) = \chi C_m \frac{\partial V_m}{\partial t} + \chi I_{ion}, \tag{2.3.9a}$$

$$\nabla \cdot (\boldsymbol{\sigma}_I \nabla V_m) + \nabla \cdot ((\boldsymbol{\sigma}_I + \boldsymbol{\sigma}_E) \nabla u_E) = 0. \tag{2.3.9b}$$

There are a number of options for the form of I_{ion}, the simplest is the phenomenological model; i.e., $I_{ion} =$ (2.2.1). More typically, a system of ODEs, such as those discussed above, is used instead. In that case, we couple the ODE system with (2.3.9) via a vector, $\mathbf{s}$, of cellular states such as gate variables and ionic concentrations. With this we can rewrite (2.3.9) as [56]:

$$\frac{\partial \mathbf{s}}{\partial t} = \mathbf{f}(t, V_m, \mathbf{s}), \tag{2.3.10a}$$

$$\nabla \cdot (\boldsymbol{\sigma}_I \nabla V_m) + \nabla \cdot (\boldsymbol{\sigma}_I \nabla u_E) = \chi C_m \frac{\partial V_m}{\partial t} + \chi I_{ion}(V_m, \mathbf{s}), \tag{2.3.10b}$$

$$\nabla \cdot (\boldsymbol{\sigma}_I \nabla V_m) + \nabla \cdot ((\boldsymbol{\sigma}_I + \boldsymbol{\sigma}_E) \nabla u_E) = 0, \tag{2.3.10c}$$

where $\mathbf{f}$ is the ODE model being used.

For these equations to have a unique solution, boundary conditions are required for u_E and V_m. To simplify the model and the boundary conditions, it is assumed that the heart is surrounded

by a non-conductive medium. This assumption implies that both the intracellular and extracellular currents cannot travel past the boundary of the heart; i.e., the normal components of both components must be zero on the boundary [56]. So the boundary conditions are:

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

where $\hat{\mathbf{n}}$ is the outward surface normal of the heart. Table 2.3 contains values of various parameters used in the bidomain equations.

Table 2.3: Parameters used in the bidomain equations [56].

C_m	1.0	μF/cm^2
χ	2000	cm^{-1}
σ_l^I	3.0	mS/cm
σ_t^I	1.0	mS/cm
σ_n^I	0.31525	mS/cm
σ_l^E	2.0	mS/cm
σ_t^E	1.65	mS/cm
σ_n^E	1.3514	mS/cm

2.3.2 Monodomain Model

Another PDE model of cardiac electrophysiology is the monodomain model. The monodomain model is a simplification of the bidomain model that is easier to analyse and less computationally demanding [56]. The monodomain model arises from the simplifying assumption $\boldsymbol{\sigma}_E = \lambda \boldsymbol{\sigma}_I$, i.e., equal anisotropy rates. This assumption allows us to eliminate $\boldsymbol{\sigma}_E$ from (2.3.9) [56]. The choice of the value of λ can determine physiological accuracy, but it is non-trivial to select a value that gives the best results for a given experiment [56].

The assumption of equal anisotropy is not supported by experimental measurements of the two conductivities [56]. This reduction in physiological accuracy means that some physiological phenomena cannot be investigated with this model [56]. However, this reduction in accuracy can lead to significant gains in feasibility: the computational cost of using the monodomain model is about one-half to one-tenth the cost of using the bidomain model, depending on the complexity of the cell model used [60].

Substituting $\boldsymbol{\sigma}_E = \lambda \boldsymbol{\sigma}_I$ into (2.3.9) gives

$$\nabla \cdot (\boldsymbol{\sigma}_I \nabla V_m) + \nabla \cdot (\boldsymbol{\sigma}_I \nabla u_E) = \chi C_m \frac{\partial V_m}{\partial t} + \chi I_{ion}, \tag{2.3.11}$$

$$\nabla \cdot (\boldsymbol{\sigma}_I \nabla V_m) + (1+\lambda) \nabla \cdot (\boldsymbol{\sigma}_I \nabla u_E) = 0. \tag{2.3.12}$$

Equation (2.3.12) implies

$$\nabla \cdot (\boldsymbol{\sigma}_I \nabla u_E) = -\frac{1}{1+\lambda} \nabla \cdot (\boldsymbol{\sigma}_I \nabla V_m). \tag{2.3.13}$$

Using (2.3.13) in (2.3.11) yields a single PDE, and rearranging some terms gives the standard formulation of the monodomain model [56]

$$\frac{\lambda}{1+\lambda} \nabla \cdot (\boldsymbol{\sigma}_I \nabla V_m) = \chi C_m \frac{\partial V_m}{\partial t} + \chi I_{ion}. \tag{2.3.14}$$

Similarly, the two boundary conditions of the bidomain model are reduced to the boundary condition

$$\hat{\mathbf{n}} \cdot (\boldsymbol{\sigma}_I \nabla V_m) = 0. \tag{2.3.15}$$

2.3.3 Torso Model

A simplifying assumption made by the bidomain model is that the heart is surrounded by a non-conductive medium. This is not a realistic assumption: the heart is surrounded by a conductive medium, the torso [56]. To model both the heart and torso, several choices must be made regarding the coupling between the heart and the surrounding tissue. This results in several different ways to model the problem [56]. The discussion in this section is restricted to the model presented in [56]. An example torso model resulting from a different set of assumptions can be found in [50].

The process beginning with Ohm's law is equation (2.3.1) can be repeated to produce an additional PDE for flow of electricity in the torso

$$\nabla \cdot (\boldsymbol{\sigma}_T \nabla u_T) = 0, \tag{2.3.16}$$

where u_T is the current in the torso and $\boldsymbol{\sigma}_T$ is the conductivity tensor for the torso.

The boundary conditions for the bidomain model came from an assumption that does not hold the torso model. Hence, the boundary conditions must be completely reformulated. The first assumption is that the extracellular domain is in direct contact with the torso, giving

$$u_E = u_T.$$

The second assumption is that the intracellular domain is completely insulated from the torso,

giving

$$\hat{\mathbf{n}} \cdot (\boldsymbol{\sigma}_I \nabla V_m + \boldsymbol{\sigma}_I \nabla u_E) = 0,$$

on the surface of the heart. This also implies that

$$\hat{\mathbf{n}} \cdot (\boldsymbol{\sigma}_I \nabla V_m + (\boldsymbol{\sigma}_I + \boldsymbol{\sigma}_E) \nabla u_E) = \hat{\mathbf{n}} \cdot (\boldsymbol{\sigma}_T \nabla u_T),$$

on the heart's surface. The final assumption is that no current flows beyond the boundary of the torso, giving the boundary condition on the torso to be

$$(\boldsymbol{\sigma}_T \nabla u_T) \cdot \hat{\mathbf{n}} = 0.$$

The complete torso model is then

$$\begin{aligned}
\nabla \cdot (\boldsymbol{\sigma}_I \nabla V_m) + \nabla \cdot (\boldsymbol{\sigma}_I \nabla u_E) &= \chi C_m \frac{\partial V_m}{\partial t} + \chi I_{ion}, && x \in H, \\
\nabla \cdot (\boldsymbol{\sigma}_I \nabla V_m) + \nabla \cdot ((\boldsymbol{\sigma}_I + \boldsymbol{\sigma}_E) \nabla u_E) &= 0, && x \in H, \\
\nabla \cdot (\boldsymbol{\sigma}_T \nabla u_T) &= 0, && x \in T, \\
u_E &= u_T, && x \in \partial H, \\
\hat{\mathbf{n}} \cdot (\boldsymbol{\sigma}_I \nabla V_m + \boldsymbol{\sigma}_I \nabla u_E) &= 0, && x \in \partial H, \\
\hat{\mathbf{n}} \cdot (\boldsymbol{\sigma}_I \nabla V_m + (\boldsymbol{\sigma}_I + \boldsymbol{\sigma}_E) \nabla u_E) &= \hat{\mathbf{n}} \cdot (\boldsymbol{\sigma}_T \nabla u_T), && x \in \partial H, \\
(\boldsymbol{\sigma}_T \nabla u_T) \cdot \hat{\mathbf{n}} &= 0, && x \in \partial T,
\end{aligned}$$

where H is the heart domain, T is the torso domain, x is any point in $H \cup T$, and $\partial H, \partial T$ are the boundaries of the heart and torso, respectively.

This research only considers the heart in isolation, and hence the torso model is not used.

2.4 Total Heart Modelling

The models discussed thus far only describe the behaviour one important aspect of cardiac physiology: electrophysiology. In order to model total heart function we must take into account some other factors as well. The first is soft tissue mechanics [26]. Electrophysiological models describe the process that controls the mechanical contraction of the heart but do not describe the contraction itself. Models of soft tissue mechanics describe this behaviour with large deformation elasticity theory [26]. The second factor in modelling total heart function is ventricular and coronary fluid mechanics, modelled using the Navier–Stokes equations [26]. Models for each of these three features of cardiac physiology can be combined to form a model of total heart function [26]. There are additional important aspects of cardiac physiology that are not generally added to models of total heart

function, such as cardiac metabolism, because, at present, there are no adequate models [26]. This research deals only with models of cardiac electrophysiology, so details of the other types of heart models are not given. It is, however, important to note that models of cardiac electrophysiology can be considered one part of a larger model.

Chapter 3

Numerical Methods

3.1 Differential Equations

The material is this section is largely adapted from [10].

A *differential equation* is an equation that contains an unknown (vector) function $\mathbf{y}(t)$ and its derivatives. If the unknown function $\mathbf{y}$ depends only on one independent variable t, then the equation

$$\mathbf{F}\left(t, \mathbf{y}, \mathbf{y}', ..., \mathbf{y}^{(n)}\right) = \mathbf{0}, \tag{3.1.1}$$

is called an *ordinary differential equation* (ODE). A *solution* of (3.1.1) is a function, $\mathbf{y}(t)$, with n derivatives that satisfy (3.1.1). When we have known values for a function and $n-1$ of its derivatives at a single point t_0, we have an *initial-value problem* (IVP). This IVP usually gives us a *unique solution* to an ODE, as opposed to a *general solution* that contains arbitrary constants of integration.

In this thesis, we are primarily concerned with ODEs of the form:

$$\frac{d\mathbf{y}}{dt} = \mathbf{f}\left(t, \mathbf{y}\right), \quad \mathbf{y}(t_0) = \mathbf{y}_0. \tag{3.1.2}$$

That is, we are looking at IVPs that describe the rate of change of $\mathbf{y}$ over time, t. Without loss of generality, we can assume $t_0 = 0$.

Sometimes we may need more than one independent variable to mathematically model a given process. For example, modelling heat flow through a metal rod requires independent variables in both time and space; the temperature in the rod depends on the position in the rod as well as time. If we have more than one independent variable, then we may take a derivative with respect to any one of the independent variables while treating the other independent variables as constants. This is called a *partial derivative*. The partial derivative of a function $\mathbf{f}$ with respect to the variable y_i is denoted $\partial\mathbf{f}/\partial y_i$. If a differential equation contains a partial derivative of an unknown function of more than one independent variable, it is called a *partial differential equation* (PDE). The *del*

operator, denoted ∇, is frequently used to simplify the specification of PDEs and is defined as

$$\nabla = \sum_{i=1}^{n} \hat{\mathbf{e}}_i \frac{\partial}{\partial y_i},$$

where $\{\ \hat{\mathbf{e}}_i : i \leq i \leq n\ \}$ is the standard Cartesian basis of $\mathbb{R}^n$.

3.2 Numerical Methods for ODEs

The material in subsections 3.2.1, 3.2.2, 3.2.3, and 3.2.4 has been mainly adapted from [52] and [27].

3.2.1 Basic Concepts

Although analytical methods exist to solve differential equations, in practice we are often faced with a differential equation that cannot be solved analytically. Systems of differential equations that model problems in all of the mathematical sciences are often large, complicated, and non-linear. When analytical methods are unavailable, one may use numerical methods to approximate the solution to a differential equation.

Arguably the simplest numerical method for approximating the solution of (3.1.2) is *Euler's method*, also known as *Forward Euler* (FE). One step of FE from $(t_{n-1}, \mathbf{y}_{n-1})$ to $(t_n, \mathbf{y}_n)$ is given by

$$\mathbf{y}_n = \mathbf{y}_{n-1} + \Delta t_n \mathbf{f}\left(t_{n-1}, \mathbf{y}_{n-1}\right), \tag{3.2.1a}$$

$$t_n = t_{n-1} + \Delta t_n, \tag{3.2.1b}$$

where $\mathbf{y}_n \approx \mathbf{y}(t_n)$. The time step $\Delta t_n := t_n - t_{n-1}$ need not be constant for each n. Recalling that the slope of the tangent line at time t for $\mathbf{y}(t)$ is given by

$$\mathbf{y}'(t) = \mathbf{f}\left(t, \mathbf{y}(t)\right),$$

we can see that FE is an approximation of the tangent line at each interval. That is, geometrically, the approximate solution produced by FE is the union of each of these approximations over the entire solution interval. So, in theory, as $\Delta t \to 0^+$, the approximation given by FE approaches the true solution. In other words, given infinite precision, the FE approximation converges to the true solution as Δt approaches zero. However, in practice we are limited by the finite amount of precision available on a given computer.

This notion of convergence is very important when discussing numerical methods. Each consis-

tent numerical method has an *order of convergence*. We say a method is of order p if

$$\mathbf{y}_n - \mathbf{y}(t_n) = O(\Delta t^{p+1}). \tag{3.2.2}$$

It is well known that FE is first-order accurate [52], which is generally too low for efficiency in practice.

3.2.2 Error and Stability

No matter how sound the numerical method used, the approximation process naturally produces some error. For example, we introduce error when we discretize our continuous equation, and further error is introduced when the solution is computed with finite precision. To obtain an acceptable approximation, a numerical method must limit all sources of error. One method to control the error is to estimate the error at each time step and then, if necessary, adjust the size of the step. If the error is too large, the step is rejected, and the solver tries again using a smaller step-size. If the error is too small, the solver can increase the size of the next step and thus increase efficiency.

An IVP is often called *stiff* if the choice of step-size Δt_n of a numerical method is determined by stability requirements rather than by accuracy requirements[1]. Generally, the step-size required for a stiff problem is much smaller than accuracy requirements dictate. In such cases, the numerical solution is typically much more accurate than required by the user. For efficiency purposes we would like to choose a step-size based only the accuracy requirements. We can understand this to some extent by considering *absolute stability theory*.

In absolute stability theory we consider the numerical approximation of the solution of

$$y' = \lambda y,\ t \geq 0,\ y(0) = y_0,$$

where λ is some complex-valued constant. The analytical solution of this equation is

$$y(t) = y_0 e^{\lambda t}, \tag{3.2.3}$$

and so we have

$$\lim_{t \to \infty} |y_n| \to 0 \tag{3.2.4}$$

if and only if the real part of λ is strictly less than zero. We say the *region of absolute stability* of the given numerical method is the set of all $\lambda \Delta t \in \mathbb{C}$ such that (3.2.4) holds and $|y_{n+1}| \leq |y_n|$. It

[1] Arguably, there is no universally accepted definition of a stiff problem, but this description of stiffness suffices for our study.

is for these $\lambda\Delta t$ that the method is absolutely stable. A special case is when the region of absolute stability contains the entire left-hand side of the complex plane. Methods with this property are called *A-stable*. With *A*-stable methods we have no restrictions on Δt due to stability (at least for the model problem (3.2.3)), and hence they are a good choice for stiff methods. If, in addition to being *A*-stable, a numerical method also satisfies

$$\lim_{z\to\infty} R(z) = 0, \tag{3.2.5}$$

where $z = \lambda\Delta t$ and

$$R(z) = 1 + z\mathbf{b}^T(\mathbf{I} - z\mathbf{A})^{-1}(1,\ldots,1)^T, \tag{3.2.6}$$

then the method is said to be *L*-stable. *L*-stability requires desirable behaviour on the far left-hand side of the complex plane, making *L*-stable methods a good choice for stiff problems.

When a numerical method is able to produce a stable approximation, we are then interested in the accuracy of the approximation. When the exact solution is not known, we may be able to generate a *reference solution*, e.g., by using a variable step-size solver with low error tolerances until two approximations are produced that agree to a desired number of significant digits. For ODE experiments in this study, we generate a reference solution by using a high-order, variable step-size implicit solver and lowering the error tolerances for successive approximations until two approximations are identical for at least 10 significant digits at N equally spaced output points $t_i = it_f/N$, $i = 1, 2, \ldots, N$, where t_f is the endpoint of solution interval and $N = 100$. We can then measure the error in the approximation, $\mathbf{y}$, relative to the reference solution, $\hat{\mathbf{y}}$. A popular way to quantify error in the literature on heart simulation is the *Relative Root Mean Squared (RRMS) error* of the transmembrane potential [24]:

$$RRMS := \sqrt{\frac{\sum_{i=1}^{N}(V_{m,i} - \hat{V}_{m,i})^2}{\sum_{i=1}^{N}\hat{V}_{m,i}^2}}, \tag{3.2.7}$$

where $V_{m,i}$ is the numerical approximation and $\hat{V}_{m,i}$ is the reference solution to V_m at time t_i as described above. Given the many other approximations made in creating the model, an RRMS error of 5% is generally considered acceptable.

As a more familiar measure of error, we also quantify error via the *global error*, which we define as

$$e_{\text{global}} := \max_i |V_{m,i} - \hat{V}_{m,i}|, \quad i = 1, 2, \ldots, N. \tag{3.2.8}$$

3.2.3 Runge–Kutta Methods

The FE method can be viewed as the simplest of a more general class of numerical methods for solving IVPs known as *Runge–Kutta* (RK) methods; see, e.g., [27]. RK methods aim to improve on the FE method by increasing the accuracy of the numerical solution by means of additional $\mathbf{f}$ evaluations (or *stages*) within a given time step. A general s-stage RK method has the form

$$\mathbf{y}_n = \mathbf{y}_{n-1} + \Delta t_n \sum_{i=1}^{s} b_i \mathbf{K}_i,$$

where for $i = 1, 2, \ldots, s$,

$$\mathbf{K}_i = \mathbf{f}\left(t_{n-1} + \Delta t_n \, c_i, \mathbf{y}_{n-1} + \Delta t_n \sum_{j=1}^{s} a_{ij} \mathbf{K}_j \right),$$

and which can be summarized via the *Butcher tableau* [27]:

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

or

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

A RK method is *explicit* if $\mathbf{A}$ is strictly lower triangular; otherwise it is *implicit*. With *explicit Runge–Kutta* (ERK) methods, the stages can be computed successively and their contributions combined to produce a high-order approximation at the end of the step. With *implicit Runge–Kutta* (IRK) methods, a (generally) non-linear system of equations must be solved at every time step for all stages simultaneously. However, because of their superior stability properties, IRK methods are well-suited for stiff problems [27].

The choice of parameters defining a specific RK method is often made based on desired order requirements. In other words, we try to pick terms such that the truncation error is of a certain order. The idea is that repeated function evaluations are used to eliminate lower-order truncation error terms. Perhaps the most popular high-order ERK method is the classical RK method, which

is a four-stage, fourth-order ERK method, and which we denote by ERK4:

$$\begin{array}{c|cccc} 0 & 0 & 0 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\ \frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ \hline & \frac{1}{6} & \frac{1}{3} & \frac{1}{3} & \frac{1}{6} \end{array}$$

Another popular ERK method is the Dormand–Prince 5(4) (DP) pair, which is of order 5 and has an auxiliary method of order 4 for error estimation and step-size control [52]. This method is the basis of the popular Matlab routine `ode45`.

The simplest IRK method, the *backward Euler* (BE) method, is a one-stage, first-order method with Butcher tableau

$$\begin{array}{c|c} 1 & 1 \\ \hline & 1 \end{array} \tag{3.2.9}$$

If an IRK method has a matrix $\mathbf{A}$ that is lower triangular, then it is called a diagonally IRK (DIRK) method. The size of the non-linear system that is required to be solved at each for each intermediate value can be reduced by using a DIRK method rather than a fully implicit method [20]. If we additionally have all a_{ii} to be equal, then we may reuse the coefficient matrix $\mathbf{I} - \Delta t a_{ii} \frac{\partial \mathbf{f}}{\partial \mathbf{y}}$ needed for the solution of non-linear equations by Newton's method and be even more efficient. Methods with this property are called *singly DIRK* (SDIRK) methods. An example of an SDIRK method is SDIRK4 of [20, p. 100]. This is an L-stable 5-stage, order-4 method with an auxiliary method of order-3 used for error estimation and step-size control. IRK methods that are not DIRK methods can have other desirable features. For example, an s-stage method from the Radau family of methods (see [20]) has order $2s-1$. This gives high-order methods with fewer stages than explicit methods or DIRK methods with the same order. A popular example of a Radau methods is the L-stable, 3-stage, order-5 Radau IIA method RADAU5 of [20, p. 74].

Three methods based upon implicit Runge–Kutta methods contained within Matlab are also considered in this thesis: `ode23s`, `ode23t`, and `ode23tb`. The first is based upon a modified Rosenbrock formula (see [20]) of order 2 and is meant for the solution of stiff problems, as an alternative to the popular `ode15s`. The second is based upon the trapezoidal rule (see [20]), and the third is based upon both the trapezoidal rule and a backward differentiation formula (see section 3.2.4). All three methods are recommended in Matlab's documentation for the solution of stiff problems with crude error tolerances. Hence, these methods may be well suited to the simulation of cardiac electrophysiological models as we have described. For further details on these methods, see Matlab's online documentation at
http://www.mathworks.com/access/helpdesk/help/techdoc/ref/ode23.html.

3.2.4 Linear Multistep Methods

Another general class of numerical methods to solve ODEs is called *Linear Multistep Methods* (LMMs). A k-step LMM computes the numerical solution using information from the last k steps. This distinguishes LMMs from RK methods, which only use information from the previous step. Let $\mathbf{f}_l = \mathbf{f}(t_l, \mathbf{y}_l)$, where $\mathbf{y}_l$ is the approximate solution at $t = t_l$. The general form of an LMM is given by

$$\sum_{j=0}^{k} \alpha_j \mathbf{y}_{n-j} = \Delta t \sum_{j=0}^{k} \beta_j \mathbf{f}_{n-j},$$

where k is the number of steps, and α_j, β_j are the coefficients that define the specific LMM. Some assumptions are made on the choice of coefficients: $\alpha_0 \neq 0$ and $|\alpha_k| + |\beta_k| \neq 0$. In the formulation above, it is also assumed that the past k integration steps are equally spaced. Often α_0 is set to 1 to eliminate arbitrary scaling.

The reliance on $k-1$ steps to compute the solution requires that a LMM must have some procedure for computing the first $k-1$ steps. This can be done by, e.g., starting the integration process with a Runge–Kutta method or starting with lower-order LMMs.

The most common LMMs are derived using polynomial interpolation. The most frequently used non-stiff LMMs, called *Adams methods*, solve (3.1.1) using

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

and using an interpolating polynomial in place of $\mathbf{f}(t, \mathbf{y}(t))$. Explicit Adams methods, called *Adams-Bashforth methods*, interpolate $\mathbf{f}$ through $t_{n-1}, \ldots, t_{n-k}$. Implicit Adams methods, called *Adams–Moulton methods*, interpolate at t_n in addition to at the other k points. The most frequently used LMMs for stiff problems are called *Backward Differentiation Formulas* (BDFs). BDFs have $\beta_j = 0$, with the exception of β_0, and are derived by interpolating over past values of $\mathbf{y}$ rather than past values of $\mathbf{f}$ and collocating the ODE at t_n.

The nature of LMMs makes them an unsuitable choice for cardiac simulation. A LMM requires the storage of state vectors for all cells in the simulation over several time steps. A RK method, on the other hand, only requires storage of one step. This becomes particularly important for large scale experiments, such as those presented in [44], for example, which used two billion cells.

3.3 The Finite Element Method

The finite element method (FEM) is used for the numerical solution of PDEs. See [7, 61] for an introduction to the subject, or see [56] for a detailed description of the FEM applied to models of

cardiac electrophysiology. To illustrate the FEM, we consider the following coupled PDEs:

$$\frac{\partial V_m}{\partial t} = \nabla \cdot (\boldsymbol{\sigma}_I \nabla V_m) + \nabla \cdot (\boldsymbol{\sigma}_I \nabla u_E), \tag{3.3.1a}$$

$$0 = \nabla \cdot (\boldsymbol{\sigma}_I \nabla V_m) + \nabla \cdot ((\boldsymbol{\sigma}_I + \boldsymbol{\sigma}_E) \nabla u_E). \tag{3.3.1b}$$

These PDEs are obtained during the solution of the bidomain model, described in section 2.3.1, via operator splitting. This solution technique is elaborated on in sections 3.4.1 and 3.5. As in Chapter 2, we solve this PDE system for V_m and u_E. In this section we outline the FEM using [7, 61] for the FEM theory, and [56] for material specific to the bidomain model.

Let S be the function space in which we seek the solution, and let Ω be the domain. The FEM begins [61] by multiplying (3.3.1) with an arbitrary function, $\psi \in S$, which is known as a *test function*, integrating over Ω, and simplifying the result using Green's Theorem [39] and the boundary conditions, (2.3.11), to get:

$$\frac{\partial}{\partial t} \int_\Omega V_m \psi \, dx = - \int_\Omega \boldsymbol{\sigma}_I \nabla V_m \cdot \nabla \psi \, dx - \int_\Omega \boldsymbol{\sigma}_I \nabla u_E \cdot \nabla \psi \, dx, \tag{3.3.2a}$$

$$0 = \int_\Omega \boldsymbol{\sigma}_I \nabla V_m \cdot \nabla \psi \, dx + \int_\Omega (\boldsymbol{\sigma}_I + \boldsymbol{\sigma}_I) \nabla u_E \cdot \nabla \psi \, dx. \tag{3.3.2b}$$

This is known as the *weak form* of (3.3.1). Any solution satisfying (3.3.1) will also satisfy the weak form (3.3.2) [7]. It can also be shown that any solution of (3.3.2) that is twice differentiable is also a solution of (3.3.1) [7].

The weak form is a continuous problem; in order to solve this problem on a computer we must discretize it. This is done by dividing Ω into a finite set of polygonal domains and then forming a finite-dimensional function space $S_h \subset S$, in which we define the numerical solution [56]. Triangles and tetrahedra are common choices for the polygons when solving 2- and 3-dimensional problems, respectively. For example, the FEM may partition Ω into a set of non-overlapping triangles. This partition forms the *mesh*. The numerical solution is found at the vertices of the polygonal region, called the *nodes*. The FEM partitions Ω into a set of m triangles. The union of these m triangles form a polygonal domain $\Omega_h \subset \Omega$, the boundary of which approximates $\partial\Omega$.

Let S_h be the function space in which we seek to find the numerical solution. Let N be the number of nodes in Ω_h, and let $\bar{x}_i, i = 1, ..., N$, be the nodes. We take S_h to be the space spanned by basis functions $\phi_i, i = 1, ..., N$, defined by positive linear functions satisfying [61]:

$$\phi_i = \begin{cases} 1, & \text{if at node } i, \\ 0, & \text{elsewhere.} \end{cases}$$

That is, ϕ_i vanishes on all triangles that do not have $\bar{x}_i$ as a vertex. It is also required that ϕ_i

decreases linearly from $\bar{x}_i$ to an adjacent vertex.

The numerical solution of the two unknown variables, V_m and u_E, can be written as a linear combination of the basis functions [61]:

$$V_m = \sum_{j=1}^{N} V_{m,j}\phi_j, \quad u_E = \sum_{j=1}^{N} u_{E,j}\phi_j, \tag{3.3.3}$$

where $V_{m,j}, u_{E,j}$ are time-dependent coefficients. We take $\psi = \phi_i$, and, at each point $\bar{x}_i$ in the domain, we obtain the following equations from (3.3.2) in the discrete space [61]:

$$\sum_{j=1}^{N} \frac{\partial}{\partial t} V_{m,j} \int_\Omega \phi_i \phi_j \, dx = -\sum_{j=1}^{N} V_{m,j} \int_\Omega \boldsymbol{\sigma}_I \nabla\phi_i \cdot \nabla\phi_j \, dx - \sum_{j=1}^{N} u_{E,j} \int_\Omega \boldsymbol{\sigma}_I \nabla\phi_i \cdot \nabla\phi_j \, dx, \tag{3.3.4a}$$

$$0 = \sum_{j=1}^{N} V_{m,j} \int_\Omega \boldsymbol{\sigma}_I \nabla\phi_i \cdot \nabla\phi_j \, dx + \sum_{j=1}^{N} u_{E,j} \int_\Omega (\boldsymbol{\sigma}_I + \boldsymbol{\sigma}_E) \nabla\phi_i \cdot \nabla\phi_j \, dx, \tag{3.3.4b}$$

for $i = 1, 2, ..., N$.

The basis functions are known, and so we need to solve for the time-dependent unknowns that define the numerical solution. Because (3.3.4) is a system of linear equations, we may rewrite it as [61, 56]:

$$\begin{bmatrix} \mathbf{M}\frac{d\mathbf{V}_m}{dt} \\ 0 \end{bmatrix} = -\begin{bmatrix} \mathbf{K}^I & \mathbf{K}^I \\ \mathbf{K}^I & \mathbf{K}^{I+E} \end{bmatrix} \begin{bmatrix} \mathbf{V}_m \\ \mathbf{u}_E \end{bmatrix}, \tag{3.3.5}$$

where

$$\begin{aligned} \mathbf{M}(i,j) &= \int_\Omega \phi_i \phi_j \, dx, \\ \mathbf{K}^I(i,j) &= \int_\Omega \boldsymbol{\sigma}_I \nabla\phi_i \cdot \nabla\phi_j \, dx, \\ \mathbf{K}^{I+E}(i,j) &= \int_\Omega (\boldsymbol{\sigma}_I + \boldsymbol{\sigma}_E) \nabla\phi_i \cdot \nabla\phi_j \, dx, \end{aligned}$$

and $\mathbf{V}_m$ and $\mathbf{u}_E$ are vectors consisting of the time-dependent coefficients, $V_{m,j}$ and $u_{E,j}$, defined above.

Equation (3.3.5) is a *differential algebraic equation* (DAE). A detailed introduction to DAEs is beyond the scope of this thesis; for such an introduction, we refer to [20]. In the case of (3.3.5), we have a linear DAE that can be viewed as an ODE subject to a constraint; the first row representing the ODE, and the second representing the constraint. In this case, we may solve (3.3.5) with the methods discussed section 3.2. This spatial discretization from the PDE to a set of DAEs is known as the *method of lines*.

3.4 Splitting Methods

When solving a system of ODEs or PDEs, it may be inefficient to use one numerical method for every part of the system [25]. For example, some components of the system may be most efficiently solved with one numerical method and other parts of the system most efficiently solved with another numerical method. Rather than solving such a system with one numerical method and accepting the consequences of inefficiency, it is often better to use a *splitting method* [25]. A splitting method uses a divide-and-conquer strategy to solve the system by breaking the system into parts that can be solved efficiently with one particular method.

3.4.1 Operator Splitting

One particular splitting technique is called *operator splitting* or *time-splitting* [25]. Consider, for example, a linear constant-coefficient ODE system:

$$\mathbf{y}'(t) = \mathbf{A}\mathbf{y}(t),\ \mathbf{y}(0) = \mathbf{y}_0. \tag{3.4.1}$$

The local solution of (3.4.1) on on $[t_n, t_{n+1}]$ is given by [25]

$$\mathbf{y}(t_{n+1}) = e^{\tau \mathbf{A}}\mathbf{y}(t_n), \tag{3.4.2}$$

where $\tau = t_{n+1} - t_n$. If we rewrite $\mathbf{A}$ as $\mathbf{A} = \mathbf{A}_1 + \mathbf{A}_2$, then, by the properties of the matrix exponential [3], we get:

$$\mathbf{y}(t_{n+1}) = e^{\tau \mathbf{A}_1} e^{\tau \mathbf{A}_2}\mathbf{y}(t_n). \tag{3.4.3}$$

Equation (3.4.3) then leads to the operator splitting method on the interval $[t_n, t_{n+1}]$ as follows. When solving (3.4.1) numerically, we divide (3.4.1) with $\mathbf{A} = \mathbf{A}_1 + \mathbf{A}_2$ into two sub-problems

$$\frac{\mathbf{y}^*(t)}{dt} = \mathbf{A}_1\mathbf{y}^*(t), \qquad \text{for } t_n < t \le t_{n+1},\ \mathbf{y}^*(t_n) = \mathbf{y}_n, \tag{3.4.4}$$

$$\frac{\mathbf{y}^{**}(t)}{dt} = \mathbf{A}_2\mathbf{y}^{**}(t), \qquad \text{for } t_n < t \le t_{n+1},\ \mathbf{y}^{**}(t_n) = \mathbf{y}^*(t_{n+1}). \tag{3.4.5}$$

These two sub-problems are solved sequentially, starting with $\mathbf{y}_n$ and ending with $\mathbf{y}_{n+1} = \mathbf{y}^{**}(t_{n+1})$. This operator splitting technique is called *Godunov splitting* and is first-order accurate. This splitting does introduce additional error at each time-step, beyond truncation and round-off error, called *splitting error*. However, it can be shown that if $\mathbf{A}_1$ and $\mathbf{A}_2$ commute, then the splitting error goes to zero [25].

Higher-order operator splitting methods are also used. The simplest second-order operator splitting technique is called *Strang splitting* [25]. Applied to (3.4.1), Strang splitting is expressed

using the notation of (3.4.3) as

$$\mathbf{y}_{n+1} = (e^{\frac{1}{2}\tau\mathbf{A}_1}e^{\frac{1}{2}\tau\mathbf{A}_2})(e^{\frac{1}{2}\tau\mathbf{A}_2}e^{\frac{1}{2}\tau\mathbf{A}_1})\mathbf{y}_n = e^{\frac{1}{2}\tau\mathbf{A}_1}e^{\tau\mathbf{A}_2}e^{\frac{1}{2}\tau\mathbf{A}_1}\mathbf{y}_n. \tag{3.4.6}$$

Naturally, a second-order operator splitting technique leads to an overall second-order accurate solution only if the numerical solution to the split equations are (at least) second-order. Both Godunov and Strang splitting fit into a more general formulation of linear operator splitting methods given by

$$\mathbf{y}_{n+1} = \sum_{i=1}^{s} \alpha_i \left(\prod_{j=1}^{r} e^{\tau\beta_{ij}\mathbf{A}_1} e^{\tau\gamma_{ij}\mathbf{A}_2} \right) \mathbf{y}_n, \tag{3.4.7}$$

where the β_{ij}, γ_{ij}, and α_i are the $s \times r$ coefficients that define the specific method [25]. For consistency, it is required that

$$\sum_{i=1}^{s} \alpha_i = 1. \tag{3.4.8}$$

An operator splitting technique in the form of (3.4.7) with order greater than two must have some negative coefficients [25]. Specifically,

$$\text{order } > 2 \text{ and } \alpha_i > 0,\ 1 \le i \le s \ \Rightarrow\ \min(\beta_{ij}) < 0 \text{ and } \min(\gamma_{ij}) < 0, \tag{3.4.9}$$

implying a negative time step for both $\mathbf{A}_1$ and $\mathbf{A}_2$ [25]. Negative time steps can have potentially troublesome consequences. For example, negative time steps can make the stability analysis of a given operator splitting method very difficult [25]. Several types of problems are ill-posed when a negative time step is applied [25], e.g., equations involving a diffusion term, such as the bidomain model.

3.4.2 Implicit-Explicit Methods

When the right-hand side of an ODE can be written as the sum of two terms

$$\frac{d\mathbf{y}}{dt} = \mathbf{f}_{IM}\left(t, \mathbf{y}\right) + \mathbf{f}_{EX}\left(t, \mathbf{y}\right), \tag{3.4.10}$$

it is often natural to consider approximating the contributions of $\mathbf{f}_{IM}(t, \mathbf{y})$ and $\mathbf{f}_{EX}(t, \mathbf{y})$ using different numerical methods. Such methods are known as *additive* methods. In general, when the right-hand side of an ODE can be written as the sum of n terms, these methods are called n-additive methods. When the constituent numerical methods are RK methods, then they are known as n-additive RK methods. Furthermore, if $d\mathbf{y}/dt = \mathbf{f}_{IM}(t, \mathbf{y})$ is such that it is best approximated with an implicit method and $d\mathbf{y}/dt = \mathbf{f}_{EX}(t, \mathbf{y})$ is such that it is best approximated with an explicit method, we may use an *implicit-explicit* (IMEX) method in an attempt to approximate

the solution to this ODE efficiently [29]. An example of when an IMEX method would be useful is when $\mathbf{f}_{IM}(t, \mathbf{y})$ consists of stiff and/or linear terms and $\mathbf{f}_{EX}(t, \mathbf{y})$ consists of non-stiff and/or non-linear terms. Again, when the constituent implicit and explicit methods are RK methods, we have an IMEX-RK method.

For example [4], the combination of the FE and BE methods (equations (3.2.1) and (3.2.9)) gives the IMEX-RK method:

$$\mathbf{y}_n = \mathbf{y}_{n-1} + \Delta t_n \left(\mathbf{f}_{EX} \left(t_{n-1}, \mathbf{y}_{n-1} \right) + \mathbf{f}_{IM} \left(t_n, \mathbf{y}_n \right) \right).$$

More generally an s-stage IRK method with coefficients $\mathbf{A}, \mathbf{c}, \mathbf{b}$ is combined with an $(s+1)$-stage ERK method with coefficients $\hat{\mathbf{A}}, \hat{\mathbf{b}}, \hat{\mathbf{c}}$. As is conventional, we assume that $\hat{\mathbf{c}} = (0, \ \mathbf{c})^T$, and the IRK method is taken to be an SDIRK method [4, 29].

One step of an IMEX-RK method is given by the following [4]. Set

$$\hat{\mathbf{K}}_1 = \mathbf{f}_{EX}(t_{n-1}, \mathbf{y}_{n-1}).$$

Then, for $i = 1, ..., s$, solve for $\mathbf{K}_i$

$$\mathbf{K}_i = \mathbf{f}_{IM} \left(t_{n-1} + \Delta t_n c_i, \mathbf{y}_i \right),$$

where

$$\mathbf{y}_i = \mathbf{y}_{n-1} + \Delta t_n \sum_{j=1}^{i} a_{ij} \mathbf{K}_j + \sum_{j=1}^{i} \hat{a}_{i+1,j} \hat{\mathbf{K}}_j.$$

Evaluate

$$\hat{\mathbf{K}}_{i+1} = \mathbf{f}_{EX} \left(t_{n-1} + \Delta t_n c_i, \mathbf{y}_i \right).$$

Finally, evaluate

$$\mathbf{y}_n = \mathbf{y}_{n-1} + \Delta t_n \sum_{j=1}^{s} b_j \mathbf{K}_j + \sum_{j=1}^{s+1} \hat{b}_j \hat{\mathbf{K}}_j.$$

In this thesis, we consider the IMEX-RK methods ARK3(2)4L[2]SA and ARK5(3)8L[2]SA from [29], which we denote by ARK3 and ARK5, respectively. ARK3 is an IMEX-RK method having 4 stages and order 3 with an auxiliary method of order 2 for error estimation and automatic step-size control; ARK5 has 8 stages, order 5, and an auxiliary method of order 4. The Butcher tableaux of ARK3 and ARK5 are listed in Appendix C of [29].

We split the ODE (3.1.2) on each time step $[t_{n-1}, t_n]$ by letting $\mathbf{f}_{IM}(t, \mathbf{y}) := \mathbf{J}(t_{n-1}, \mathbf{y}_{n-1})\mathbf{y}(t)$ and $\mathbf{f}_{EX}(t, \mathbf{y}) := \mathbf{f}(t, \mathbf{y}) - \mathbf{J}(t_{n-1}, \mathbf{y}_{n-1})\mathbf{y}(t)$, where $\mathbf{J} := \partial \mathbf{f} / \partial \mathbf{y}$. We note that this splitting is such that only the linear term is treated implicitly, and hence there is no need for a Newton iteration when solving the implicit equations.

3.5 Popular Numerical Methods for Cardiac Electrophysiology Simulation

3.5.1 Methods for Single Cell Simulations

An alternative to using classical methods such as FE that is popular in the cardiac simulation literature is commonly called the Rush–Larsen (RL) method [51], even though it dates back to work done earlier by Hodgkin and Huxley [23]. The RL method is an example of a *non-standard finite difference* (NSFD) method. NSFD methods use mixtures of discretizations in order to produce methods that preserve specific properties of the exact solution, such as positivity or asymptotic correctness. The RL method advances the solution to the gating equations (2.2.11) using

$$y_n = y_\infty + (y_{n-1} - y_\infty)e^{-\frac{\Delta t_n}{\tau_y}}, \tag{3.5.1}$$

which represents the exact solution of (2.2.11) *assuming all variables besides y are constant*. FE is then used to advance the solution of the remaining equations. Using this method the Luo–Rudy model, for example, is no longer stiff [55]; i.e., the time step-size can be chosen based on accuracy considerations. However, this method is only first-order accurate and thus suffers from the usual drawbacks of low-order methods. RL can be put in a general class of methods called *waveform relaxation* (WR) methods. WR methods break up the solution of a system of ODEs over a time interval such that each part may be solved separately for the complete time interval. See [27] and the references therein for a brief introduction to WR methods.

Combinations of (3.5.1) with methods other than FE were investigated in [37]. In particular, both the combination of (3.5.1) with a second-order SDIRK method and the combination of (3.5.1) with the second-order implicit midpoint method were demonstrated to be second-order. Numerical experiments involving the models of Courtemanche et al. and Winslow et al. presented in [37] demonstrated gains in efficiency relative to both FE and to (3.5.1) and FE when using these two combinations, particularly for the model of Courtemanche et al.

Other numerical methods used to solve the ODEs in such models include SDIRK methods, fully implicit RK methods, and multi-step methods based on BDFs; see, e.g., [57].

3.5.2 Methods for Multiple Cell Simulations

The use of splitting methods for cardiac simulations is quite common, particularly for simulations in which the ODEs for myocardial cell models are coupled with PDEs describing the flow of electricity through myocardial tissue.

A second-order accurate operator splitting method for the monodomain model was studied by

Qu and Garfinkel in [47]. This method divides the solution of the monodomain PDE and the coupled ODE model into the three steps using Strang splitting (see section 3.4). A step from t_n to $t_n + \Delta t$ consists of:

Step 1. Integrate

$$\frac{\lambda}{1+\lambda}\nabla \cdot (\boldsymbol{\sigma}_I \nabla V_m) = \chi C_m \frac{\partial V_m}{\partial t} \tag{3.5.2}$$

for a step of size $\Delta t/2$ using the solution at time t_n as the initial condition.

Step 2. Integrate the cell model a step of size Δt using the solution of Step 1 as the initial condition.

Step 3. Integrate (3.5.2) for a step of size $\Delta t/2$ using the solution of Step 2 as the initial condition.

The advantage of this splitting method is that it transforms the non-linear PDE problem into a linear PDE and a set of non-linear ODEs. The authors use RL together with FE to solve the ODEs and FE to solve the PDE. The authors also investigate the use of an alternating direction implicit method [25] to solve the PDE as an alternative to FE. The use of first-order methods means that the computed solution is not second-order accurate. However, this splitting method could obtain second-order accuracy assuming more appropriate time integration methods are used; i.e., methods of at least second-order.

Note that it is possible to split the monodomain model in the same manner and instead use first-order Godunov splitting. In this case, *Step 1* would instead require a step of size Δt, and *Step 3* would not be required. Both Godunov and Strang splitting are frequently used in cardiac simulations.

Other work has been done to apply operator splitting to the monodomain model in the same manner as done by Qu and Garfinkel. For example, Yung [70] studied the monodomain model coupled with the model of Winslow et al. Yung used a BDF solver to integrate the stiff ODE model and an explicit second-order Runge–Kutta method for PDE component. Yung compared the splitting method to FE and found cases such that each method is the most efficient. Specifically, FE is the most efficient method when the membrane kinetics of the model of Winslow et al. are modified by imposing a maximum value on dP_{O_1}/dt and dP_{O_2}/dt, equations (A.3.2) and (A.3.3), and the operator splitting method is the most efficient method when the model of Winslow et al. is not modified. Yung also demonstrates that the operator splitting method is unsuitable for the non-stiff FHN model due to the large amount of error in the computed solution when compared to FE or ERK4. Trangenstein [63] studies operator splitting for the monodomain model (as well as adaptive mesh refinement) using an SDIRK method for the ODE model with a Crank–Nicolson method for the PDEs. When compared to solving the problem unsplit with the LSODE solver [22], Trangenstein's method is shown to be much more efficient. However, comparisons were only performed with a very small mesh in 1D.

The same idea for a splitting method has been applied to the bidomain model. Instead of solving (3.5.2), we solve two linear PDEs:

$$\frac{\partial V_m}{\partial t} = \nabla \cdot (\boldsymbol{\sigma}_I \nabla V_m) + \nabla \cdot (\boldsymbol{\sigma}_I \nabla u_E), \tag{3.5.3a}$$

$$0 = \nabla \cdot (\boldsymbol{\sigma}_I \nabla V_m) + \nabla \cdot ((\boldsymbol{\sigma}_I + \boldsymbol{\sigma}_E) \nabla u_E). \tag{3.5.3b}$$

This has been used by many authors. For example, Whiteley [68] investigates the use of a first-order semi-implicit method for solving the PDE component, comparing it to a first-order explicit method and a second-order Crank–Nicolson method. Whiteley shows that the semi-implicit method is slower than the explicit method when the same step-size is used for each method, but the semi-implicit method can be up to 35 times faster when the largest possible step-sizes are used. Whiteley notes that the Crank–Nicolson method takes about the same amount of execution time as the semi-implicit method. Whiteley demonstrates unphysical oscillations in the solutions produced by the semi-implicit and Crank–Nicolson methods when larger step-sizes are used. Whiteley shows that if a step-size small enough to eliminate these oscillations is selected, then the semi-implicit method is still an order of magnitude faster than the explicit method. This splitting method is also used by Han and Ng with first- and second-order IMEX schemes in [21] for both the monodomain and bidomain models. An implicit method is used for the PDE(s), and an explicit method is used to integrate the ODEs [41]. A similar approach is used by Pennacchio and Simoncini in [42],where the IMEX method forward-backward Euler [4] is used, although the focus is on solving the algebraic system arising from the discretization with forward-backward Euler.

Splitting is used in an analogous manner for the torso model by Sundnes et al. in [58]. Godunov and Strang splitting were both a part of the algorithm investigated by the authors. With Godunov splitting, BE is used to solve the split PDE system, and with Strang splitting, Crank–Nicolson is used. The choice of the splitting and integration method was dependent on a parameter in their algorithm. In both cases, a third-order implicit Runge–Kutta scheme was used for the split ODE model. Second-order convergence is demonstrated for the Strang splitting method, although a very fine mesh is required to observe this for most the complex cell model studied in the paper. The two methods are very close in terms of execution time, but the Strang splitting method is more accurate. In particular, the propagation velocity is reproduced much more accurately with the Strang splitting method; the authors argue the increased accuracy is a crucial feature for many practical applications.

Some work has been done to apply operator splitting in this manner to even more complicated models. For example, Thorvaldsen et al. [62] uses this operator splitting method for a model of cardiac electrophysiology that includes the mechanical function of the heart. As such models are

beyond the scope of this thesis, work in this area is not elaborated on.

See Chapter 3 of [56] for a detailed description of this operator splitting method applied to the monodomain, bidomain, and torso models.

Although splitting in the manner of Qu and Garfinkel has been used frequently, other splitting methods have been used as well. For example, some authors split the bidomain model even further by additionally splitting the two PDEs; see, e.g., [33]. Splitting of this type is typically limited to first-order [58]. Another approach, by Keener and Borgar [28], splits using the eigenvalues of the Jacobian of the cell model, $\mathbf{J}$. Using the fact that $\mathbf{J}$ is bounded, they find a matrix $\mathbf{A}$ such that $\mathbf{J} + \mathbf{A}$ is always positive. Using $\mathbf{A}$, they split the right hand side of the cell model, $\mathbf{f}(t, \mathbf{y})$, into $\mathbf{f}(t, \mathbf{y}) + \mathbf{A}y$ and $(-\mathbf{A}y)$. The former is solved explicitly and the latter implicitly.

Other techniques for improving the accuracy and efficiency of cardiac simulations have been studied as well, such as adaptive mesh refinement [6], parallel computing [44], and algebraic preconditioning [43]. Because the focus of this thesis is on time integration, these subjects are not elaborated on. The interested reader may consult [66] for a review work studying the solution of the bidomain model.

Chapter 4

Results

We performed two sets of numerical experiments, testing ideas to make cardiac simulation more efficient: one set of single cell experiments, and one set of two-dimensional experiments with tens of thousands of cells. Results of the first set of experiments were published in [54] and results of the second set of experiments are under review for publication as [13].

The first set of experiments we performed tested the execution time of IMEX-RK methods, described in section 3.4.2, relative to some commonly used numerical methods. As outlined in section 3.5.1, a number of numerical methods for solving single cell models of cardiac electrophysiology have been studied. This includes standard explicit methods, such as FE and ERK4, standard implicit methods, such as SDIRK or RADAU methods, and methods specialized to cardiac electrophysiology, such as the RL method. While some of these methods can be efficient relative to one another, there are disadvantages to each of them that can limit their efficiency. The explicit methods have severe step-size restrictions, and the implicit methods require expensive Newton iterations. The most popular method, RL, can also face step-size restrictions, particularly with relatively stiff models as demonstrated in [37]. IMEX-RK methods can avoid step-size restrictions seen in explicit methods, and the particular IMEX-RK methods used in this thesis do not require Newton iterations. Hence, we studied the use of an IMEX-RK method as an alternative. In this experiment our hypothesis was: *If we use an IMEX-RK method to solve a single cell cardiac electrophysiological model, then the simulation will be more efficient than if we had selected one of frequently used methods instead.* Full details and results of the experiment are presented in section 4.1.

The second set of experiments we performed studied the solution of (3.5.3), the split linear PDE component of the bidomain model. As described in section 3.5, second-order methods for this task can better reproduce key physiological properties, such as conduction velocity, when compared to first-order methods. Furthermore, the most popular second-order method for solving this model, Crank-Nicolson (CN), has relatively poor error-damping properties that often result in solutions with unphysical oscillations, such as those exhibited in [68]. We expected the L-stability property, defined in section 3.2.2, to be relevant in suppressing unphysical oscillations due to its strong damping properties. SDIRK methods are arguably the simplest possible L-stable methods. To overcome these problems and fully realize the potential of second-order splitting methods for the

bidomain equations, we studied a second-order method based on Strang splitting and a second-order L-stable SDIRK method to solve (3.5.3). We compared the SDIRK method to the CN, as well as to the simplest L-stable method, BE. Our hypothesis for this experiment was: *If we use a second-order L-stable SDIRK method to solve (3.5.3), then the solution will be free of unphysical oscillations and we will be able to obtain a solution more efficiently.* Full details and results of the experiment are presented in section 4.2.

4.1 Single Cell Experiments

4.1.1 Overview of Experiments

To evaluate the efficiency of IMEX-RK methods, we performed numerical experiments with the FE, ERK4, RL, DP, BE, SDIRK4, RADAU5, `ode23s`, `ode23t`, `ode23tb`, ARK3, and ARK5 methods. Four different cardiac electrophysiological models were used: Luo–Rudy, Courtemanche et al., Winslow et al., and Puglisi–Bers, described in sections 2.2.2, 2.2.3, 2.2.4 and 2.2.5.

Initial values for the experiments depended on the model for the particular experiment. Except for V_m, the initial values of the variables were taken to be those that correspond to the heart in its resting state. For the models of Luo–Rudy and Winslow et al. , the initial values of V_m were chosen to produce the effect of an explicit stimulus current. That is, the initial values used for V_m are above the thresholds for the cells to fire. For the models of Courtemanche et al. and Puglisi–Bers, the initial values for V_m were taken as their resting values, and an explicit stimulus was applied as given by (2.2.10). In particular, for the model of Courtemanche et al., $I_{st} = -2000$ pA/pF from $t = 0$ to $t = 2$ms and 0 elsewhere; for the Puglisi–Bers model, $I_{st} = -10$ μA/μF from $t = 1$ to $t = 5$ ms and 0 elsewhere. The specific initial values used for V_m are listed in Table 4.1. See [34, 11, 69, 45] for complete listings of the remaining initial values. These values may also be accessed from http://www.cellml.org/models/ under the directories luo_rudy_1991_version04, courtemanche_ramirez_nattel_1998_version02, winslow_rice_jafri_marban_ororke_1999_version01, and puglisi_bers_2001_version01, respectively.

Table 4.1: Initial values for $V_m = V_{rest}$.

Model	V_{rest}
Luo–Rudy	−35.0
Courtemanche et al.	−81.2
Winslow et al.	−35.0
Puglisi–Bers	−85.5

The models were solved over time intervals representing one cardiac cycle. Different time intervals were used due to specific physiological properties of the mammalian heart that each model

represents. Accordingly, the Luo–Rudy model was solved on the interval [0,450] ms, the model of Courtemanche et al. was solved on the interval [0,500] ms, the model of Winslow et al. was solved on the interval [0,300] ms, and the Puglisi–Bers model was solved on the interval [0,330] ms.

All the numerical experiments were performed on an Athlon 64 3000+ 1.8 GHz processor with 1 GB RAM. CPU times reported are the minimum of 5 runs. We note that the runs for FE, ERK4, RL, DP, ARK3, and ARK5 were performed within the `odeToJava` framework. Presently, `odeToJava` supports only ERK and linearly implicit IMEX-RK methods; numerical experiments involving other implicit techniques were performed as follows. The runs for `ode23s`, `ode23t`, and `ode23tb` were performed within Matlab. The runs for BE were performed using `ode15s` within Matlab with `MaxOrder` set to 1 and `BDF` to `'On'`. In these cases a conversion factor was determined to compare CPU time within Matlab to CPU time within `odeToJava`. This was done by computing the average of the ratios of a number of runs of `ode45` with the DP class within `odeToJava` for different tolerances. Similarly, the SDIRK4 and RADAU5 methods were run using their respective Fortran codes, and a conversion factor for CPU times was calculated by comparing runs of `DOPRI5.f` [19, p. 477] with the DP class in `odeToJava`. Conversion factors were double-checked by comparing them to conversion factors obtained with forward Euler code written in `odeToJava`, Matlab, and Fortran. All CPU times reported below for BE, `ode23s`, `ode23t`, `ode23tb`, SDIRK4, and RADAU5 reflect this conversion.

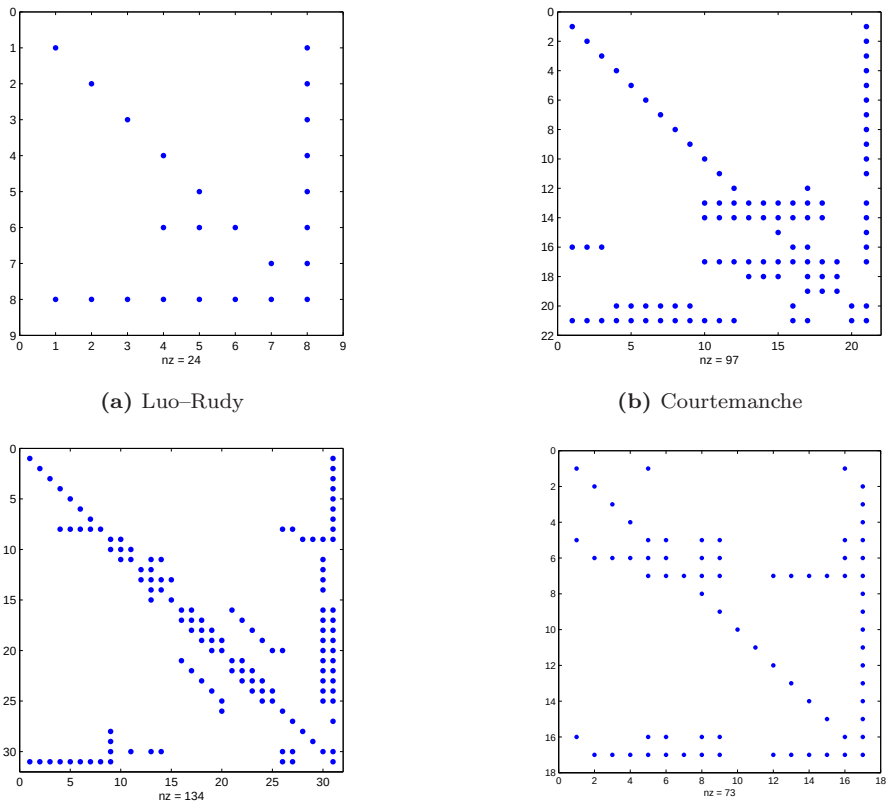
4.1.2 Customized Linear System Solver

With the splitting employed, a linear system involving $\mathbf{J}(t, \mathbf{y})$ must be solved at each time step of an IMEX-RK integration. Accordingly, we are interested in the *sparsity pattern* of $\mathbf{J}(t, \mathbf{y})$ across the entire solution interval of an ODE. A sparsity pattern can be thought of as a map of a matrix describing which entries of a matrix are always zero and which entries can be non-zero. The sparsity patterns of $\mathbf{J}(t, \mathbf{y})$ for each of the 4 models were generated; see Figure 4.1 for the sparsity patterns of the four models. If an element of $\mathbf{J}(t, \mathbf{y})$ is always zero, it may be possible to omit it during Gaussian Elimination [15], the method used for solving linear systems by our implicit solver. This means that we may further optimize the IMEX-RK results compared to the previous section by customizing a Gaussian Elimination code to take advantage of these sparsity patterns. Results of the ARK3 and ARK5 methods with customized Gaussian Elimination routine are included below.

4.1.3 Constant Step Size Results

In Table 4.2, we report the maximum step-size, $\Delta t_{\max}$, to 3 significant figures, for which FE, ERK4, and RL produce an approximation with less than 5% RRMS error. We also report the corresponding CPU times required, the RRMS error, and the global error.

For FE and ERK4, $\Delta t_{\max}$ is also the step-size that produces a stable solution, indicating that



(a) Luo–Rudy

(b) Courtemanche

(c) Winslow

(d) Puglisi–Bers

Figure 4.1: Sparsity patterns for the four models.

Table 4.2: Results for the constant step-size FE, ERK4, and RL methods.

		$\Delta t_{\max}$	CPU Time (s)	RRMS Error	Global Error
L	FE	1.34E–2	2.24E–1	3.08E–2	2.81E+0
R	ERK4	1.86E–2	6.77E–1	3.87E–2	3.51E+0
	RL	2.50E–1	**4.29E–2**	4.79E–2	5.39E+0
C	FE	1.94E–2	8.05E–1	2.30E–3	1.94E+0
R	ERK4	2.68E–2	2.30E+0	4.73E–2	6.73E+0
T	RL	3.45E–1	**7.89E–2**	4.97E–2	3.75E+1
W	FE	1.07E–4	4.04E+1	7.78E–8	9.91E–2
I	ERK4	1.30E–4	1.31E+2	3.40E–2	2.61E–1
N	RL	2.80E–4	**2.25E+1**	4.86E–2	6.08E+0
P	FE	1.08E–2	4.54E–1	5.20E–3	3.54E–1
B	ERK4	1.48E–2	1.08E+0	1.30E–2	1.33E–1
	RL	4.30E–1	**6.50E–2**	4.83E–2	8.54E+0

Table 4.3: Variable step-size results for the Luo–Rudy model.

	TOL	CPU Time (s)	Sparse CPU Time (s)	RRMS Error	Global Error	Average Δt
DP	–2	5.20E–1	–	1.85E–2	2.41E+0	9.26E–2
BE	–3	1.45E–1	–	1.61E–2	2.92E+0	1.08E+0
SDIRK4	–1	4.80E–2	–	1.77E–2	2.05E+0	1.25E+1
RADAU5	–1	7.30E–2	–	6.75E–4	1.78E–2	1.25E+1
`ode23s`	–1	**1.03E–2**	–	3.25E–2	5.82E+0	1.50E+1
`ode23t`	–1	1.17E–2	–	2.97E–2	6.60E+0	1.00E+1
`ode23tb`	–1	1.04E–2	–	1.17E–2	2.37E+0	1.25E+1
ARK3	–2	4.00E–2	1.80E–2	5.33E–3	1.91E+0	7.25E+0
ARK5	–2	3.40E–2	2.40E–2	7.46E–4	2.42E+0	1.15E+1

these methods generally view the problems as stiff. Thus the resulting RRMS errors can be well below the desired levels. We also see that FE takes approximately 2-3 times less CPU time than ERK4 on all 4 models studied. Hence, as is well known, higher order does not lead to greater efficiency when non-stiff methods are applied to stiff problems with moderate accuracy requirements.

We note that care must be exercised when determining $\Delta t_{\max}$ for the RL method because the RRMS error produced is not a monotonically increasing function of the step-size. In other words, there exist $\Delta t < \Delta t_{\max}$ for which the RRMS error *exceeds* 5%. We see that the RL method is the most efficient for all four models. It ranges from about 2 times faster for the model of Winslow et al. to about 10 times faster for the CRT model.

4.1.4 Variable Step-Size Tests

Tables 4.3–4.6 respectively report the results from DP, BE, SDIRK4, RADAU5, `ode23s`, `ode23t`, `ode23tb`, ARK3, and ARK5 with variable step-sizes applied to each of the 4 cardiac electrophysiological models. We run the models using standard error estimation and step-size control algorithms (see, e.g., [52]) for a range of absolute and relative tolerances. We set absolute tolerances equal to relative tolerances and define TOL to be their logarithm to base 10; e.g., TOL= -3 implies both absolute and relative tolerances were set to 10^{-3}. Integer values of TOL were run from -1 to -6 for all solvers and all 4 models, but details are reported for only the runs with the best CPU time that met the 5% RRMS error criterion.

For the Luo–Rudy (LR) model, ARK3 and ARK5 outperform seven of the other methods in the study even without taking sparsity into account. With sparsity, ARK3 is about 1.7 times slower than the method that produces an acceptable solution in the least amount of time, `ode23s`.

Table 4.4: Variable step-size results for the model of Courtemanche et al.

	TOL	CPU Time (s)	Sparse CPU Time (s)	RRMS Error	Global Error	Average Δt
DP	–3	6.37E–1	–	5.26E–4	2.71E–1	1.66E–1
BE	–2	8.26E–2	–	2.40E–2	1.21E+1	2.34E+0
SDIRK4	–2	1.28E–1	–	7.51E–4	1.91E–1	9.43E+0
RADAU5	–1	2.40E–1	–	2.61E–3	2.79E+0	1.04E+1
`ode23s`	–1	4.52E–1	–	3.13E–2	2.50E+1	1.25E+1
`ode23t`	–1	3.04E–1	–	7.66E–3	1.16E+0	8.33E+0
`ode23tb`	–1	3.16E–1	–	6.87E–3	3.55E+0	1.02E+1
ARK3	–2	6.90E–2	**4.50E–2**	7.87E–3	1.06E+0	7.14E+0
ARK5	–2	1.37E–1	6.00E–2	3.60E–2	2.28E+1	1.19E+1

Table 4.5: Variable step-size results for the model of Winslow et al.

	TOL	CPU Time (s)	Sparse CPU Time (s)	RRMS Error	Global Error	Average Δt
DP	–3	3.00E+1	–	7.92E–3	4.15E–1	4.88E–3
BE	–4	9.51E–1	–	3.71E–2	1.25E+1	1.32E–1
SDIRK4	–2	2.72E–1	–	5.03E–3	5.81E–1	3.75E+0
RADAU5	–3	6.13E–1	–	7.68E–3	9.84E–1	2.65E+0
`ode23s`	–1	**3.90E–2**	–	1.85E–2	4.94E+0	5.17E+0
`ode23t`	–2	4.72E–2	–	3.80E–2	9.76E+0	2.03E+0
`ode23tb`	–2	**3.90E–2**	–	2.31E–2	5.27E+0	2.88E+0
ARK3	–3	2.68E–1	1.97E–1	1.46E–2	1.84E+0	1.69E+0
ARK5	–3	4.65E–1	2.92E–1	3.15E–2	3.96E+0	3.06E+0

Table 4.6: Variable step-size results for the Puglisi–Bers model.

	TOL	CPU Time (s)	Sparse CPU Time (s)	RRMS Error	Global Error	Average Δt
DP	–2	1.53E+0	–	4.43E–2	1.30E+0	4.09E–2
BE	–3	1.74E–1	–	5.69E–3	7.62E+0	6.13E–1
SDIRK4	–3	1.36E–1	–	8.48E–3	5.85E+0	4.92E+0
RADAU5	–2	1.53E–1	–	4.77E–2	5.53E+1	6.34E+0
`ode23s`	–2	6.38E–2	–	2.03E–2	8.23E+0	4.71E+0
`ode23t`	–2	3.84E–2	–	1.31E–2	6.84E+0	3.51E+0
`ode23tb`	–2	3.36E–2	–	1.44E–2	7.56E+0	4.52E+0
ARK3	–4	6.90E–2	4.60E–2	2.38E–3	5.04E–1	4.12E+0
ARK5	–3	1.12E–2	**7.00E–3**	1.89E–2	1.63E+1	1.48E+0

For the model of Courtemanche et al. (CRT), ARK3 produces an acceptable solution in the least amount of CPU time. With sparsity, ARK3 is 1.8 times faster than RL, its next closest commonly used competitor.

For the model of Winslow et al. (WIN), ARK3 and ARK5 outperform seven of the other methods in the study. With sparsity, ARK3 is about 5 times slower than the methods that produce an acceptable solution in the least amount of time, `ode23s` and `ode23tb`.

Finally, for the Puglisi–Bers (PB) model, we see that ARK5 produces an acceptable solution in the least amount of CPU time, with or without taking sparsity into account. In this case, ARK5 with sparsity produces an acceptable result about 5 times faster than `ode23tb`, its next closest competitor.

Due to the small execution time required for solving a single cycle with individual cell models, the execution time of 100 runs as described above was computed; i.e. the execution time required for 100 cardiac cycles. This was to ensure the reliability of the results by eliminating the possibility of noise influencing the results. Ratios between execution times for individual methods remained the same to at least three significant figures. In other words, results presented above for one cardiac cycle held for one hundred cardiac cycles.

4.1.5 Constant Step-Size IMEX

We also investigated the use of constant step-size ARK3 and ARK5 methods. We find that these constant step-size implementations significantly underperform the variable step-size implementation. They also significantly underperform the RL method on the LR, CRT, and PB models and the FE method on the WIN model. We omit further details.

4.2 Two-Dimensional Experiments

4.2.1 Overview of Experiments

Using `PyCC` [38], a Python based problem solving environment for the solution of PDEs, we studied the solution of the bidomain model. In particular, we focused on the split linear PDE system obtained when solving the bidomain model using operator splitting. We compared the performance of the second-order SDIRK method outlined above to a second-order implicit method, CN, and a first-order L-stable implicit method, BE. In the following, we describe the methods used in this study.

We describe a general formulation based on a Θ fractional-step method, where the Strang and Godunov splitting methods, defined in section 3.4.1, are obtained as special cases for $\Theta = 1/2$ and $\Theta = 1$, respectively. One time step requires the separate solution of two systems: one is the split cell model

$$\frac{\partial \mathbf{s}}{\partial t} = \mathbf{f}(t, V_m, \mathbf{s}), \tag{4.2.1a}$$

$$\frac{\partial V_m}{\partial t} = -I_{ion}(V_m, \mathbf{s}), \tag{4.2.1b}$$

and the other is the linear PDE system

$$\frac{\partial V_m}{\partial t} = \nabla \cdot (\boldsymbol{\sigma}_I \nabla V_m) + \nabla \cdot (\boldsymbol{\sigma}_I \nabla u_E), \tag{4.2.2a}$$

$$0 = \nabla \cdot (\boldsymbol{\sigma}_I \nabla V_m) + \nabla \cdot ((\boldsymbol{\sigma}_I + \boldsymbol{\sigma}_E) \nabla u_E). \tag{4.2.2b}$$

One step of the splitting method to advance from time t_n to time $t_{n+1} = t_n + \Delta t$ involves the solution of the two systems (denoted A and B) in three phases.

1. Using as initial conditions the solution at time t_n, solve System A for $t_n < t \leq t_{n+\Theta} := t + \Theta \Delta t$.
2. Using the solution of phase 1 as the initial condition, solve System B for $t_n < t \leq t_{n+1}$.
3. Using the solution of phase 2 as the initial condition, solve System A for $t_{n+\Theta} < t \leq t_{n+1}$.

In principle, either system (4.2.1) or (4.2.2) may be used as System A, with the other as System B. In practice, however, the quality of a numerical solution may vary significantly for different choices of A and B. As is the case in all of the literature of which we are aware, in this thesis we use the BE method and CN to solve (4.2.2) as system B. This was important for the quality of the solution produced by BE. When using the SDIRK method, we solve (4.2.2) as system A. This produced higher-quality solutions, especially for large Δt. We also note that except for the first

and last time steps, phase 3 from time step n is combined in practice with phase 1 from time step $n+1$ for efficiency; i.e., a single step from $t = t_{n+\Theta}$ to $t = t_{n+1+\Theta}$ is taken.

The algorithm used for both BE and CN can be seen as special cases of a more general method. This method, called the θ-rule, uses BE in the case of $\theta = 1$ and a combination of CN and implicit midpoint in the case of $\theta = 1/2$. When applied to (4.2.2) this method results in the following linear system to be solved at each step in time:

$$\begin{bmatrix} \mathbf{M} + \theta\Delta t\mathbf{K}^I & \Delta t\mathbf{K}^I \\ \Delta t\mathbf{K}^I & \frac{\Delta t}{\theta}\mathbf{K}^{I+E} \end{bmatrix} \begin{bmatrix} V_m^{n+1} \\ u_E^{n+\theta} \end{bmatrix} = \begin{bmatrix} (\mathbf{M} - (1-\theta)\Delta t\mathbf{K}^I)V_m^n \\ -\frac{\Delta t(1-\theta)}{\theta}\mathbf{K}^I V_m^n \end{bmatrix}, \tag{4.2.3}$$

where $u_E^{n+\theta}$ is numerical solution of u_E at time $t_n + \theta\Delta t$. When solving with CN, we used $\Theta = 1/2$, and when solving with BE we used $\Theta = 1$; i.e., we took $\Theta = \theta$.

This linear system is too large and sparse to be solved efficiently with a direct method. Instead, we used a conjugate gradient iterative solver; see, e.g., [18]. To solve a linear system with a conjugate gradient method, the system must be symmetric. Hence, the second row of (4.2.3) was scaled by Δt to meet this requirement. In all experiments, the iteration is deemed to have converged when 2-norm of the initial residual has decreased by 5 orders of magnitude; this is the default behaviour in `PyCC`.

The SDIRK method we used in this study is the L-stable, two-stage, second-order SDIRK method defined by the Butcher tableau

$$\begin{array}{c|cc} \gamma & \gamma & 0 \\ 1 & (1-\gamma) & \gamma \\ \hline & (1-\gamma) & \gamma \end{array},$$

with $\gamma = (2 - \sqrt{2})/2$ [20]. When solving with the SDIRK method we used $\Theta = 1/2$.

For a general initial-value problem, (3.1.2), this method takes one step in time by means of the iteration

$$\begin{aligned} \mathbf{Y}_1 - \Delta t\gamma\mathbf{f}(t_n + \gamma\Delta t, \mathbf{Y}_1) &= \mathbf{y}_n, \\ \mathbf{y}_{n+1} - \Delta t\gamma\mathbf{f}(t_{n+1}, \mathbf{y}_{n+1}) &= \mathbf{y}_n + \Delta t(1-\gamma)\mathbf{f}(t_n + \gamma\Delta t, \mathbf{Y}_1). \end{aligned}$$

This iteration has been simplified using the fact that $\mathbf{y}_{n+1} = \mathbf{Y}_2$.

This leads to two linear systems that need to be solved to take one step in time. First, we must solve

$$\begin{bmatrix} \mathbf{M} + \gamma\Delta t\mathbf{K}^I & \gamma\Delta t\mathbf{K}^I \\ \gamma\Delta t\mathbf{K}^I & \gamma\Delta t\mathbf{K}^{I+E} \end{bmatrix} \begin{bmatrix} V_m^* \\ u_E^* \end{bmatrix} = \begin{bmatrix} \mathbf{M}V_m^n \\ 0 \end{bmatrix},$$

to find the stage values $\mathbf{Y}_1 = (V_m^*, u_E^*)$. The second row has again been scaled, this time by $\gamma\Delta t$,

to meet the need for a symmetric system. Second, we must solve

$$\begin{bmatrix} \mathbf{M}+\gamma\Delta t\mathbf{K}^I & \gamma\Delta t\mathbf{K}^I \\ \gamma\Delta t\mathbf{K}^I & \gamma\Delta t\mathbf{K}^{I+E} \end{bmatrix} \begin{bmatrix} V_m^{n+1} \\ u_E^{n+1} \end{bmatrix} = \begin{bmatrix} \mathbf{M}V_m^n - (1-\gamma)\Delta t(\mathbf{K}^I V_m^* + \mathbf{K}^I u_E^*) \\ 0 \end{bmatrix}, \quad (4.2.4)$$

to find the approximations for V_m^{n+1} and u_E^{n+1}. These systems are also solved with the conjugate gradient solver using the same convergence criterion discussed above.

For the time discretization of the cell model ODEs (4.2.1), we use an explicit Runge–Kutta–Fehlberg embedded 4(3) pair [16] with adaptive time steps within the overall time step Δt of the splitting method. The relative and absolute tolerances in all the experiments reported here are set to 10^{-8} and 10^{-5}, respectively, which are the default values in `PyCC`. Less stringent tolerances did not reduce the overall execution times by more than a few per cent, and more stringent ones generally increased execution times. In both cases, the overall errors were unchanged to 3 digits.

4.2.2 Order of Convergence

Because extremely fine spatial and temporal resolutions are required to produce a reference solution, for convergence testing we consider a simplified problem with a restricted circular geometry. We assume the solution is rotationally invariant, depending only on the the radius r. To generate a one-dimensional reference solution for a rotationally invariant circular problem we first recall the mapping between polar coordinates and Cartesian coordinates

$$\begin{aligned} x &= r\cos\phi, \\ y &= r\sin\phi, \\ r &= (x^2+y^2)^{1/2}, \\ \phi &= \arctan\frac{y}{x}. \end{aligned}$$

Using this transformation the rotationally invariant bidomain equations are

$$\begin{aligned} r\frac{\partial V_m}{\partial t} + rI_{ion} &= \frac{\partial}{\partial r}\left(\boldsymbol{\sigma}_I r\frac{\partial V_m}{\partial r}\right) + \frac{\partial}{\partial r}\left(\boldsymbol{\sigma}_I r\frac{\partial u_E}{\partial r}\right), \\ 0 &= \frac{\partial}{\partial r}\left(\boldsymbol{\sigma}_I r\frac{\partial V_m}{\partial r}\right) + \frac{\partial}{\partial r}\left((\boldsymbol{\sigma}_I+\boldsymbol{\sigma}_E) r\frac{\partial u_E}{\partial r}\right). \end{aligned}$$

See, e.g., [60] for full details.

Reference solutions were generated for this model coupled with two cell models, the FHN model, discussed in section 2.2.1, and the model of Winslow et al., discussed in section 2.2.4. For the transmembrane potentials, the initial condition for the FHN model is

Table 4.7: Reference solution parameters.

Parameter	Value	Units
$\boldsymbol{\sigma}_I$	1.5	mS/cm
$\boldsymbol{\sigma}_E$	1.0	mS/cm
Δx	0.000625	cm
Δt	0.001	ms

$$V_m(r,0) = \begin{cases} -20 \text{ mV}, & r \leq 0.2 \text{ cm}, \\ -80 \text{ mV}, & r > 0.2 \text{ cm}, \end{cases}$$

and for the model of Winslow et al. it is

$$V_m(r,0) = \begin{cases} -20 \text{ mV}, & r \leq 0.2 \text{ cm}, \\ -95.87 \text{ mV}, & r > 0.2 \text{ cm}. \end{cases}$$

Parameters common to both models are listed in Table 4.7. In both cases, we use constant scalar conductivities with values taken from [31]. Following [60], the integration is performed using second-order Strang splitting ($\Theta = 1/2$) with CN to solve (4.2.2) and constant values $\Delta r = 1/1600$ cm and $\Delta t = 0.001$ ms. Further refinements indicate that the solution has converged to approximately 8 figures on this mesh. The final time for each model is chosen such that an action potential has begun at each of the points in the circle where a stimulus was applied. Due to the differing upstroke durations of the models, these times are 10.0 ms for the FHN model and 2.0 ms for the model of Winslow et al.

Numerical experiments were performed to determine the order of convergence using the SDIRK method instead of CN to solve (4.2.2). Similar to [60], the spatial domain for our numerical experiments was the unit circle ($0 < r \leq 1$). An initial value of $V_m = -20$ mV is applied at all nodes within a circle centred at $(0,0)$ with radius 0.2 cm. The time step-size Δt and the mesh spacing Δr are halved from one experiment to the next. The error is computed for each experiment by comparing it to the reference solution using the L_2 norm. We then compute the order of convergence using

$$\alpha = \frac{\log(\epsilon_1/\epsilon_2)}{\log(\Delta t_1/\Delta t_2)},$$

where Δt_1 and Δt_2 are two successive step-size choices and ϵ_1 and ϵ_2 are the corresponding errors.

Tables 4.8 and 4.9 demonstrate the expected second-order convergence results when using the SDIRK method. The times at which the convergence is measured have been chosen such that all the points in the stimulus circle have just generated an action potential. Convergence results for the CN and BE methods for a similar situation were presented in [59], so they are omitted here. We note that finer time and spatial steps must be used before second-order convergence is observed

Table 4.8: Convergence results for the FHN model with errors computed at $t = 10.0$ ms.

Δt	Δx	L_2 error	α
1/8	0.430	1.32E–1	–
1/16	0.215	3.38E–2	1.96
1/32	0.108	8.61E–3	1.97
1/64	0.054	2.14E–3	2.01

Table 4.9: Convergence results for the model of Winslow et al. with errors computed at $t = 2.0$ ms.

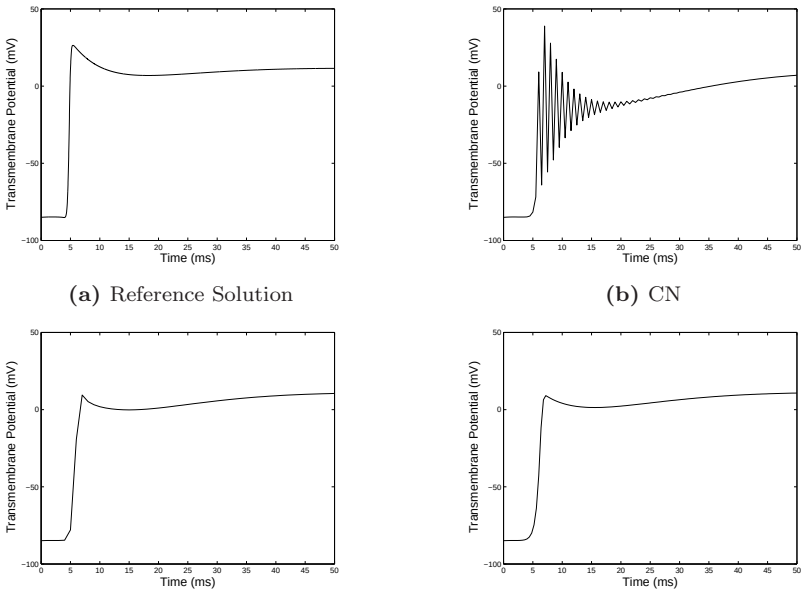
Δt	Δr	L_2 error	α
1/16	0.215	5.77E–1	–
1/32	0.108	1.70E–1	1.76
1/64	0.054	4.12E–2	2.05
1/128	0.027	9.91E–3	2.06

for the model of Winslow et al. relative to the FHN model. This can be attributed to the much faster upstroke in the model of Winslow et al. This quick upstroke leads to a sharper wavefront that requires more mesh points in both time and space to resolve accurately. In both cases, the results using the SDIRK method are qualitatively similar to those obtained using CN in [59].

4.2.3 Numerical Experiments and Results

Unphysical oscillations can be seen in the solution produced using CN for the following scenario, which is similar to an experiment described in [68]. The spatial domain is a square with 1 cm edges discretized uniformly with $N = 10\,201$ nodes and 20 000 triangles for a spatial resolution of $\Delta x = 0.01$ cm. We use conductivities 2.63 mS/cm along the fibre in both the intracellular and extracellular conductivity tensors, 0.263 mS/cm perpendicular to the fibre in intracellular conductivity tensor, and 1.087 mS/cm in extracellular conductivity tensor. For this experiment we used the LR model, discussed in section 2.2.2, as the cell model; we note that unphysical oscillations were observed in all but the simplest cell models with which we have experimented. We use this model for our experiment because it produces particularly dramatic oscillations. A stimulus is applied to the lower left-hand corner of the square, causing an excitation wave to spread across the square. For comparison purposes, we generate a reference solution with CN with $\Delta x = 0.001$ cm and $\Delta t = 0.001$ ms.

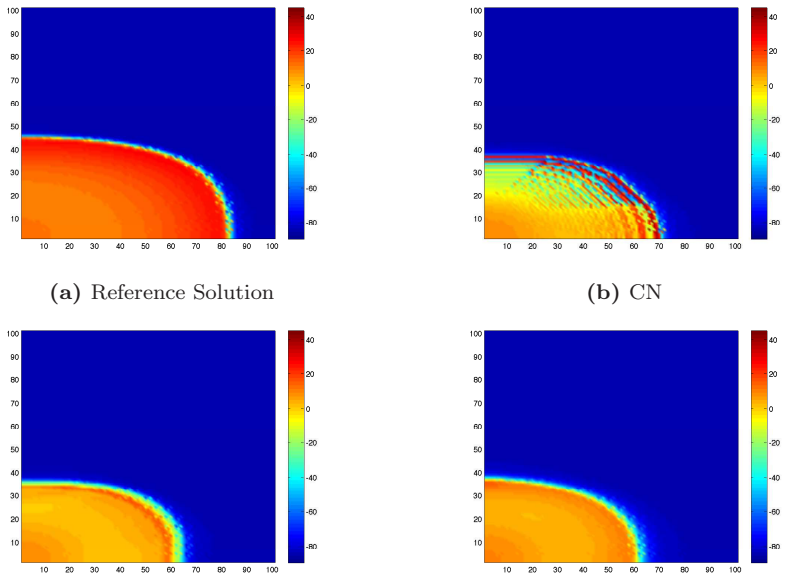
The oscillations in the solution produced using CN at a particular spatial point are demonstrated in Figure 4.2b. These oscillations are attenuated during the plateau phase, at which point the solution looks more physically reasonable. In Figure 4.3b, the solution using CN is displayed over



(a) Reference Solution **(b)** CN

(c) SDIRK **(d)** BE

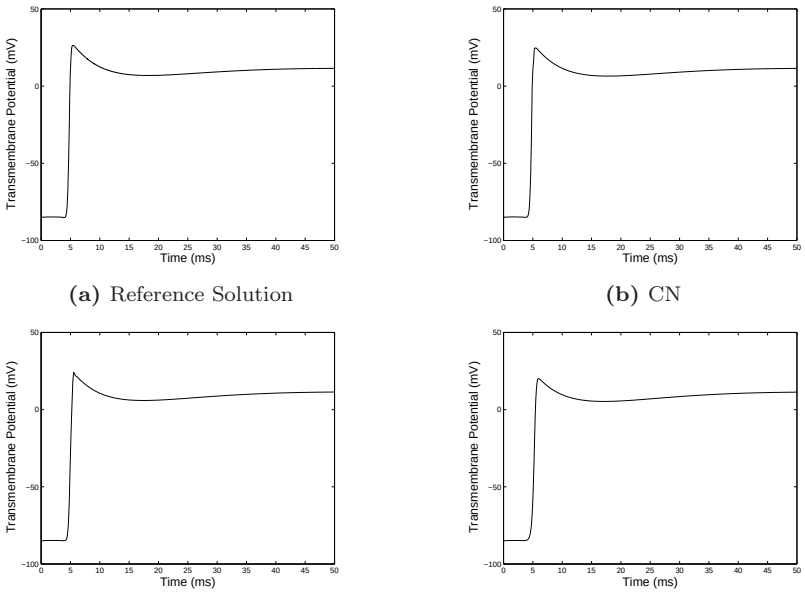
Figure 4.2: Plot of the transmembrane potential at $(0.25, 0.25)$ using $\Delta t = 0.5, 0.55, 0.4$ ms for CN, SDIRK, and BE, respectively.



(a) Reference Solution **(b)** CN

(c) SDIRK **(d)** BE

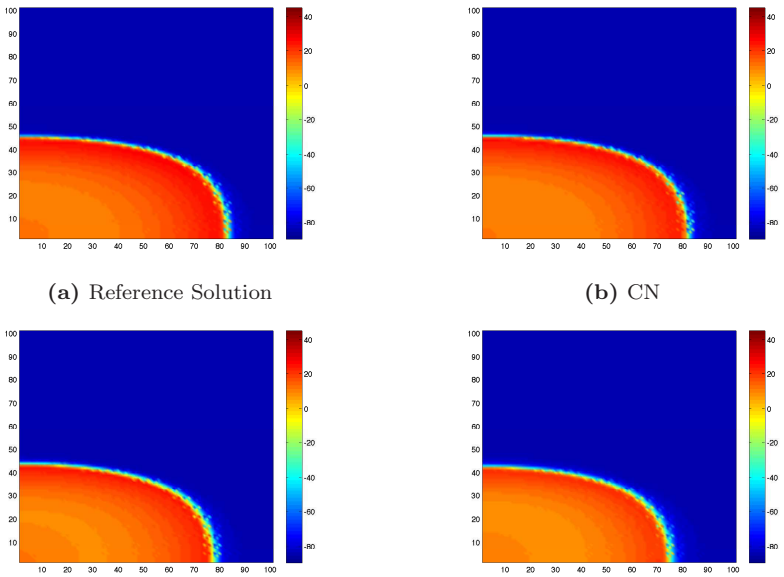
Figure 4.3: Plot of the transmembrane potential at $t = 10$ ms using $\Delta t = 0.5, 0.55, 0.4$ ms for CN, SDIRK, and BE, respectively.



(a) Reference Solution **(b)** CN

(c) SDIRK **(d)** BE

Figure 4.4: Plot of the transmembrane potential at $(0.25, 0.25)$ using $\Delta t = 0.125, 0.15, 0.1$ ms for CN, SDIRK, and BE, respectively.



(a) Reference Solution **(b)** CN

(c) SDIRK **(d)** BE

Figure 4.5: Plot of the transmembrane potential at $t = 10$ ms using $\Delta t = 0.125, 0.15, 0.1$ ms for CN, SDIRK, and BE, respectively.

the whole domain. Unphysical oscillations can be seen across the action potential wavefront. The corresponding plots for SDIRK and BE can be seen in Figures 4.2c, 4.3c, 4.2d, and 4.3d, respectively. Comparing these solutions to the reference solution presented in Figures 4.2a and 4.3a, there is some obvious error, but it is also clear that neither SDIRK nor BE exhibit any unphysical oscillations.

4.2.4 Comparison of CN, SDIRK, and BE

To compare the performance of the CN, SDIRK, and BE methods, two metrics are used to evaluate the quality of the solution. The first metric is a *weighted RRMS* error defined by

$$\epsilon_{\text{RRMS}} = \frac{\sum_i \Delta\bar{A}_i \, RRMS_i}{\sum_i \bar{A}_i}, \tag{4.2.5}$$

where $RRMS_i$ is the RRMS error, (3.2.7), at a point i in the domain, and $\Delta\bar{A}_i$ is the average area of the triangles containing point i. We consider a solution acceptable if $\epsilon_{\text{RRMS}} < 5\%$. This may generally be considered to be a rather stringent error tolerance from a computational engineering point of view; however such tolerances may be necessary to obtain meaningful data.

The second metric is based on a physiological feature of interest, the conduction velocity, which is computed as follows. The time at which top right-hand corner exceeds the threshold -10 mV was recorded. The conduction velocity is then given by the distance between the two corners divided by the recorded time.

Table 4.10: Weighted RRMS errors computed with (4.2.5).

Method	Coarse Δt	Fine Δt
CN	3.025E–1	2.72E–2
SDIRK	2.647E–1	9.16E–2
BE	2.649E–1	1.413E–1

Solutions obtained with the same computational cost using SDIRK and BE are also given in Figures 4.2 and 4.3; i.e., for each method, we find a step-size that requires approximately the same amount of CPU time to perform the simulation. Experiments were performed using a variety of CPU times, but we only present results for two sets of step-sizes to illustrate the two distinct cases we observed. The first is a set of coarse time steps, and the second is a set of fine time steps. In the case of the set of coarse time steps, we use $\Delta t = 0.5$ ms for CN, $\Delta t = 0.55$ ms for SDIRK, and $\Delta t = 0.4$ ms for BE. Solutions obtained for these combinations of a time step and method are displayed in Figures 4.2 and 4.3. In the case of the set of fine time steps, we use $\Delta t = 0.125$ ms for CN, $\Delta t = 0.15$ ms for SDIRK, and $\Delta t = 0.1$ ms for BE. In this case CN does not produce noticeable unphysical oscillations. Solutions obtained for these combinations of a time step and method are displayed in Figures 4.4 and 4.5.

Weighted RRMS errors computed with (4.2.5) are presented in Table 4.10. In the case of the set

of coarse time steps, CN produces more error than both SDIRK and BE. SDIRK and BE produce about the same amount of error, despite the larger step used by SDIRK. However, in all three cases the error is larger than 5%. So, although SDIRK can eliminate unphysical oscillations, it is unable to satisfy the 5% weighted RRMS tolerance. In the case of the set of the set of fine time steps, CN is the most accurate method. Comparing CN and SDIRK, we see that SDIRK has about three times more error for the same amount of computation time. Comparing CN and BE, we see that BE has about five times more error for the same amount of computation time. Hence the efficiency advantages of second-order methods over first-order methods remain present.

Table 4.11: Conduction velocity in mm/s. Note that the conduction velocity of the reference solution is 505 mm/s.

Δt	CN	SDIRK	BE
1	–	288	220
1/2	336	344	325
1/4	475	445	379
1/8	491	479	426

Table 4.12: Conduction velocity in mm/s.

Δt	CN	SDIRK	BE
Coarse Time Steps	336	347	343
Fine Time Steps	491	469	439

The results for conduction velocity are presented in Tables 4.11 and 4.12. Table 4.11 shows the relationship between conduction velocity and time step. This relationship has been shown before (see, e.g., [46]) and is presented here only to give some context to the results in Table 4.12. Table 4.12 gives the conduction velocity for the time steps used above. For fine time steps, CN is able to most accurately capture the conduction velocity. For coarse time steps, SDIRK is able to most accurately capture the conduction velocity. In other words, the most accurate method with respect to (3.2.7) is also the most accurate method with respect to conduction velocity. The large difference between the conduction velocity for the set of coarse time steps and the reference conduction velocity helps to explain the large error seen for all three methods. The slow conduction velocity means that there are large errors at each node between the edge of the wavefront in the numerical solution and the edge of the wavefront in the reference solution.

We have tested the robustness of the performance of CN in other experiments designed to increase the stiffness. These included lowering the conductivities by an order of magnitude to reflect the lowest values we have observed in the literature, using more realistic irregular domains, and using the much stiffer cell model of Winslow et al. Although in these experiments we found the differences in the performance of the CN and SDIRK were made smaller, CN still produced a

superior solution in all cases. Because these experiments did not present any different or additional conclusions, we omit further discussion.

Overall, the general conclusion is that for a given amount of execution time, the SDIRK method produces the most accurate solutions for coarse error tolerances ($\epsilon_{\text{RRMS}} > 5\%$), and CN produces the most accurate solutions for stringent error tolerances ($\epsilon_{\text{RRMS}} \leq 5\%$). In other words, the bidomain model is mildly stiff under the conditions investigated here.

Chapter 5

Conclusions and Future Work

5.1 Conclusions

We present conclusions for the two sets of experiments in sections 5.1.1 and 5.1.2.

5.1.1 Single Cell Experiments

We compared the performance of several numerical methods for approximating solutions to ODEs found in four popular mathematical models of cardiac electrical activity. In particular, we compared the performance two IMEX-RK methods (ARK3 and ARK5) to other commonly used numerical methods for these models, i.e., FE, ERK4, RL, `ode23s`, `ode23t`, `ode23tb`, DP, SDIRK4, and RADAU5.

For constant step-sizes, the RL method is the most efficient for all four models. It ranges from being approximately two times faster than the FE method for the WIN model to approximately 5–10 times faster than FE for the other models.

For variable step-sizes, the ARK methods outperformed the Dormand–Prince method for all four models. We obtained qualitatively similar results from a comparison of the ARK methods with the Bogacki–Shampine 3(2) method, which is the underlying method behind Matlab's `ode23` routine; we do not comment on this further.

Overall, a variable step-size implementation of ARK3 or ARK5 with a customized linear system solver was the most efficient numerical method for the CRT and PB models. For the LR and WIN models, both ARK3 and ARK5 outperform seven of the ten other numerical methods but do not outperform `ode23s`, `ode23t`, and `ode23tb`. For these two models, `ode23s` was the most efficient of all the numerical methods studied. For the WIN model, `ode23tb` was as efficient as `ode23s`.

The results in this experiment indicate that it is generally advisable to use a numerical method with an inexpensive implicit component and implemented with variable step-sizes and specialized techniques that take advantage of specific problem structure. When variable step-sizes are not possible, the RL method is the most efficient method.

The results of this experiment only considered ODE models of one cell. Further investigations of models involving large numbers of cells coupled with PDEs in two or three dimensions is nec-

essary to fully establish the potential of the results presented here. In particular, linearly implicit IMEX operator-splitting methods that do not rely on constant step-sizes may lead to substantial performance gains.

5.1.2 Two-Dimensional Experiments

We have investigated numerical methods within a commonly used operator splitting technique for solving the bidomain model. Specifically we have considered a two-stage, second-order, L-stable SDIRK method to solve the split linear PDE system (4.2.2) as an alternative to the popular CN and BE methods. The second-order convergence of using the SDIRK method was demonstrated for two cases: the bidomain model coupled with the FHN model and the model of the Winslow et al. Unphysical oscillations were produced when using CN; the use of the L-stable SDIRK or BE methods was shown to eliminate such oscillations. In particular, for coarse time steps (moderate error tolerances), the use of CN produced noticeable unphysical oscillations and about 1.2 times more error than SDIRK for the same amount of computation time. For fine time steps (stringent error tolerances), the use of CN produced a more accurate solution than both SDIRK and BE for the same amount of computation time. In our experiments, the solution obtained using the SDIRK method contained about three times more error than that produced by using CN. We confirmed that in both cases the method with the lowest weighted RRMS error (4.2.5) was also the method that best approximated the conduction velocity.

Overall, our experiments indicate that in two-dimensional simulations with a weighted RRMS error tolerance of 5%, the use of second-order operator splitting with the CN method delivers the most accurate solutions for a given amount of computation time. That is, we did not find that the stronger damping properties of the SDIRK method provided a computational advantage at these tolerances, implying that under these conditions the bidomain model is only mildly stiff.

5.2 Future Work

This work opens up several directions that can be investigated from this point.

- *Investigate the efficiency of IMEX-RK methods for solving PDE models.*

  Some preliminary work for this step was completed in [12], which looked at the solution of the monodomain model coupled with the LR model. ARK5 was compared to the operator splitting method of Qu and Garfinkel [47], and it was demonstrated that ARK5 can be up to 4 times faster. The comparison was done with a combination of Comsol Multiphysics, Matlab, and `odeToJava`. Unfortunately, this combination was unable to support anything more than a very small mesh and, consequently, it was unclear how the results would scale to mesh sizes more frequently seen in practice. Hence, the results are not entirely clear and

as such are not reported on further. It does, however, show that this is a direction worthy of further investigation.

Due to the troubles with the combination of Comsol Multiphysics, Matlab, and `odeToJava`, a different framework for these numerical experiments is needed. As an alternative, a study using `propag` is proposed. This is software for large scale simulation of depolarization and repolarization in the human heart developed at l'Université de Montréal. Some preliminary work has been done to implement an IMEX-RK solver inside of this software package to complete this task.

- *Optimizing IMEX-RK methods for solving cardiac electrophysiological models.*

  We have only investigated the use of two particular IMEX-RK methods but it may be possible to design even better IMEX-RK methods. It would be interesting to see if it is possible to improve on these results with a more suited IMEX-RK technique, ideally with one that is designed with a particular model in mind. This would begin with designing methods for single cell models but it could be extended for the PDE models in the long term.

  Similarly, the choice of splitting is not necessarily optimal. It might be fruitful to examine other ways to split the terms in cardiac electrophysiological models when using an IMEX-RK method.

- *Implement and study a new second-order operator splitting method.*

  The stronger damping properties of the SDIRK method did not provide an advantage for the experiments presented in this thesis. However, more demanding simulations (for example, using unstructured grids on realistic highly irregular 3D geometries) may increase the stiffness of the model and ultimately favour the use of SDIRK. With minimal programming effort, experiments could be performed in 3D similar to those presented in this thesis in 2D. So this is a natural next step.

References

[1] R. R. Aliev and A. V. Panfilov. A simple two-variable model of cardiac excitation. *Chaos, Solitions and Fractals*, 7(3):293–301, 1996.

[2] C. Antzelevitch and A. Burashnikov. Cardiac arrthymias: Reentry and triggered activity. In N. Sperelakis, Y. Kurachi, A. Terzic, and M. V. Cohen, editors, *Heart Physiology and Pathophysiology*. Academic Press, fourth edition, 2001.

[3] V. I. Arnol'd. *Ordinary differential equations*. Springer-Verlag, Berlin, 1992.

[4] U. M. Ascher, S. J. Ruuth, and R. J. Spiteri. Implicit-explicit Runge-Kutta methods for time-dependent partial differential equations. *Appl. Numer. Math.*, 25(2–3):151–167, 1997.

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

[6] Y. Belhamadia. A time-dependent adaptive remeshing for electrical waves of the heart. *IEEE Trans. Biomed. Eng.*, 55(2):443–452, 2008.

[7] S. C. Brenner and L. R. Scott. *The Mathematical Theory of Finite Element Methods*. Springer, Berlin, 1994.

[8] F. J. L. Van Capelle and D. Durrer. Computer simulation of arrhythmias in a network of coupled excitable elements. *Circ. Res.*, 47:454–466, 1980.

[9] C. J. Clements. Nonlinear wave propagation in an anisotropic medium. Master's thesis, Dalhousie University, 1996.

[10] E. A. Coddington. *An introduction to ordinary differential equations*. Prentice-Hall Inc., Englewood Cliffs, 1961.

[11] 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.

[12] R. C. Dean. Effectiveness of implicit-explicit Runge-Kutta methods for the solution of cardiac electrophysiological models. CMPT 880 project, 2007.

[13] R. C. Dean, R. J. Spiteri, and J. Sundnes. Suppressing unphysical oscillations in the bidomain model. 2008. Preprint.

[14] J. A. DiMasi, R. W. Hansen, and H. G. Grabowski. The price of innovation: new estimates of drug development costs. *J. Health Econ.*, 22(2):151–185, 2003.

[15] I. S. Duff, A. M. Erisman, and J. K. Reid. *Direct methods for sparse matrices*. The Clarendon Press Oxford University Press, New York, second edition, 1989.

[16] E. Fehlberg. Low-order classical Runge–Kutta formulas with stepsize control and their application to some heat transfer problems. Technical report, NASA, 1969.

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

[18] G. H. Golub and C. F. Van Loan. *Matrix computations*. Johns Hopkins University Press, Baltimore, MD, third edition, 1996.

[19] E. Hairer, S. P. Nørsett, and G. Wanner. *Solving ordinary differential equations. I.* Springer-Verlag, Berlin, second edition, 1993.

[20] E. Hairer and G. Wanner. *Solving Ordinary Differential Equations II: Stiff and Differential-Algebraic Problems*. Springer-Verlag, Berlin, second edition, 1996.

[21] Y. Han and K. T. Ng. Implicit-explicit methods for nonlinear bidomain modeling. *Ann. Biomed. Eng.*, 28(Suppl. 1):34–34, 2000.

[22] A. C. Hindmarsh. ODEPACK, a systematized collection of ODE solvers. In *Scientific computing (Montreal, Que., 1982)*, IMACS Trans. Sci. Comput., I, pages 55–64. IMACS, New Brunswick, NJ, 1983.

[23] 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.

[24] B. M. Horacek, J. W. Warren, P. Stovicek, and C. L. Feldman. Diagnostic accuracy of derived versus standard 12-lead electrocardiograms. *J. Electrocardio.*, 33(Suppl):155–160, 2000.

[25] W. Hundsdorfer and J. Verwer. *Numerical solution of time-dependent advection-diffusion-reaction equations*. Springer-Verlag, Berlin, 2003.

[26] P. J. Hunter, A. J. Pullan, and B. H. Smaill. Modeling total heart function. *Annu. Rev. Biomed. Eng.*, 5:147–177, 2003.

[27] A. Iserles. *A first course in the numerical analysis of differential equations*. Cambridge University Press, Cambridge, 1996.

[28] J. P. Keener and K. Bogar. A numerical method for the solution of the bidomain equations in cardiac tissue. *Chaos*, 8(1):234–241, 1998.

[29] C. A. Kennedy and M. A. Carpenter. Additive Runge-Kutta schemes for convection-diffusion-reaction equations. *Appl. Numer. Math.*, 44(1–2):139–181, 2003.

[30] A. G. Kléber and Y. Rudy. Basic mechanisms of cardiac impulse propagation and associated arrhythmias. *Physiol. Rev.*, 84(2):431–488, 2004.

[31] R. N. Klepfer, C. R. Johnson, and R. S. Macleod. The effects of inhomogeneities and anisotropies on electrocardiographic fields: a 3-d finite-element study. *Biomedical Engineering, IEEE Transactions on*, 44(8):706–719, Aug 1997.

[32] B. Y. Kogan, W. J. Karplus, B. S. Billett, A.T. Pang, H. S. Karagueuzian, and S. S. Kahn. The simplified FitzHugh–Nagumo model with action potential duration restitution: effects on 2-D wave propagation. *Physica D*, 50:327–340, 1991.

[33] G. Lines. *Simulating the Electrical Activity in the Heart - A Bidomain Model of the Ventrciles Embedded in a Torso*. PhD thesis, Department of informatics, University of Oslo, 1999.

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

[35] C. Luo and Y. Rudy. A dynamic model of the cardiac ventricular action potential. i. simulations of ionic currents and concentration changes. *Circ. Res.*, 74(6):1071–1096, 1994.

[36] C. Luo and Y. Rudy. A dynamic model of the cardiac ventricular action potential. ii. afterdepolarizations, triggered activity, and potentiation. *Circ. Res.*, 74(6):1097–1113, 1994.

[37] M. C. MacLachlan, J. Sundnes, and R. J. Spiteri. A comparison of non-standard solvers for ODEs describing cellular reactions in the heart. *Comput. Methods. Biomech. Biomed. Engin.*, to appear.

[38] K.-A. Mardal, O. Skavhaug, G. T. Lines, G. A. Staff, and Å. Ødegård. Using python to solve partial differential equations. *Computing in Science and Engineering*, 2007.

[39] J. E. Marsden and A. J. Tromba. *Vector Calculus*. W. H. Freeman and Company, New York, 2003.

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

[41] K. T. Ng. personal communication, 2007.

[42] M. Pennacchio and V. Simoncini. Efficient algebraic solution of reaction-diffusion systems for the cardiac excitation process. *J. Comput. Appl. Math.*, 145(1):49–70, 2002.

[43] G. Plank, M. Liebmann, R. W. dos Santos, Edward J. Vigmond, and Gundolf Haase. Algebraic multigrid preconditioner for the cardiac bidomain model. *IEEE Trans. Biomed. Eng.*, 54(4):585–596, 2007.

[44] M. Potse and A. Vinet. Large-scale integrative modeling of the human heart. In *HPCS 2008*, 2008.

[45] 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.

[46] A. J. Pullan, M. L. Buist, and L. K. Cheng. *Mathematically Modelling the Electrical Activity of the Heart: From Cell to Body Surface and Back Again*. World Scientific, New Jersey, 2005.

[47] Z. Qu and A. Garfinkel. An advanced algorithm for solving partial differential equation in cardiac conduction. *IEEE Trans. Biomed. Eng.*, 46(9):1166–1168, 1999.

[48] D. Randell, W. Burggren, and K. French. *Eckert Animal Physiology*. W. H. Freeman and Company, New York, fourth edition, 1997.

[49] J. M. Rogers and A. D. McCulloch. A collocation–galerkin finite element model of cardiac action potential propagation. *IEEE Trans. Biomed. Eng.*, 41(8):743–757, 1994.

[50] B. J. Roth. A comparison of two boundary conditions used with the bidomain model of cardiac tissue. *Ann. Biomed. Eng.*, 19(6):669–687, 1991.

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

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

[53] K. Skouibine and W. Krassowska. Increasing the computational efficiency of a bidomain model of defibrillation using a time-dependent activating function. *Ann. Biomed. Eng.*, 28(7):772–780, 2000.

[54] R. J. Spiteri and R. C. Dean. On the performance of an implicit-explicit Runge–Kutta method in models of cardiac electrical activity. *IEEE Trans. Biomed. Eng.*, 55(5):1488–1495, May 2008.

[55] R. J. Spiteri and M. C. MacLachlan. An efficient non-standard finite difference scheme for an ionic model of cardiac action potentials. *J. Difference Equ. Appl.*, 9(12):1069–1081, 2003.

[56] 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.

[57] J. Sundnes, G. T. Lines, and A. Tveito. ODE solvers for a stiff system arising in the modeling of the electrical activity of the heart. *Int. J. Nonlinear Sci.*, 4(1):67–80, 2002.

[58] 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. *Math. Biosci.*, 194(2):233–248, 2005.

[59] 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(2):233–248, 2005.

[60] J. Sundnes, B. F. Nielsen, K.-A. Mardal, X. Cai, G. T. Lines, and A. Tveito. On the computational complexity of the bidomain and the monodomain models of electrophysiology. *Ann. Biomed. Eng.*, 34(7):1088–1097, 2006.

[61] V. Thomée. *Galerkin Finite Element Methods for Parabolic Problems*. Springer-Verlag, Berlin, 1984.

[62] T. Thorvaldsen, H. Osnes, J. Sundnes, and A. McCulloch. An operator splitting technique for integrating cardiac electro-mechanics. 2006. Preprint.

[63] J. A. Trangenstein and C. Kim. Operator splitting and adaptive mesh refinement for the Luo-Rudy I model. *J. Comput. Phys.*, 196(2):645–679, 2004.

[64] L. Tung. *A Bi-domain model for Describing Ischemic Myocardial D-C Potentials*. PhD thesis, Massachusetts Institute of Technology, 1978.

[65] K. H. W. J. Ten Tusscher and A. V. Panfilov. Cell model for efficient simulation of wave propagation in human ventricular tissue under normal and pathological conditions. *Phys. Med. Biol.*, 51(23):6141–6156, 2006.

[66] E. J. Vigmond, R. W. dos Santos, A. J. Prassl, M. Deo, and G. Plank. Solvers for the cardiac bidomain equations. *Progress Biophys. Mol. Biol.*, In Press, 2007.

[67] N. Virag, J. M. Vesin, and L. Kappenberger. A computer model of cardiac electrical activity for the simulation of arrhythmias. *Pacing Clin. Electrophysiol.*, 21(11 Part 2):2366–2371, 1998.

[68] J. P. Whiteley. An efficient numerical technique for the solution of the monodomain and bidomain equations. *IEEE Trans. Biomed. Eng.*, 53(11):2139–2147, 2006.

[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] C. K. Yung. Application of a stiff, operator-splitting scheme to the computational modelling of electrical propagation in cardiac ventricles. Master's thesis, The Johns Hopkins University, 2000.

Appendix A

Mathematical Models

A.1 The Luo–Rudy model

Inward currents
Fast sodium current

$$I_{\mathrm{Na}} = \overline{G}_{\mathrm{Na}} \cdot m^3 \cdot h \cdot j \cdot (V_m - E_{\mathrm{Na}}) \tag{A.1.1}$$

Activation gate, m

$$\frac{dm}{dt} = \alpha_m(1-m) - \beta_m m \tag{A.1.2a}$$

$$\alpha_m = \frac{0.32(V_m + 47.13)}{1 - e^{-0.1(V_m+47.13)}} \tag{A.1.2b}$$

$$\beta_m = 0.08e^{-V_m/11} \tag{A.1.2c}$$

Fast inactivation gate, h

$$\frac{dh}{dt} = \alpha_h(1-h) - \beta_h h \tag{A.1.3a}$$

$$\alpha_h = \begin{cases} 0.135e^{(V_m+80)/-6.8} & V_m < -40\mathrm{mV} \\ 0 & V_m \geq -40\mathrm{mV} \end{cases} \tag{A.1.3b}$$

$$\beta_h = \begin{cases} 3.56e^{0.079V_m} + 3.1 \cdot 10^5 e^{0.35V_m} & V_m < -40\mathrm{mV} \\ \dfrac{1}{0.13(1 + e^{(V_m+10.66)/-11.1})} & V_m \geq -40\mathrm{mV} \end{cases} \tag{A.1.3c}$$

Slow inactivation gate, j

$$\frac{dj}{dt} = \alpha_j(1-j) - \beta_j j \tag{A.1.4a}$$

$$\alpha_j = \begin{cases} \dfrac{-1.2714 \cdot 10^5 e^{0.2444V_m} - 3.474 \cdot 10^{-5} e^{-0.04391V_m} \cdot (V_m + 37.78)}{1 + e^{0.311(V_m+79.23)}} & V_m < -40\mathrm{mV} \\ 0 & V_m \geq -40\mathrm{mV} \end{cases} \tag{A.1.4b}$$

$$\beta_j = \begin{cases} \dfrac{0.1212e^{-0.01052V_m}}{1 + e^{-0.1378(V_m+40.14)}} & V_m < -40\mathrm{mV} \\ \dfrac{0.3e^{-2.535 \cdot 10^{-7} V_m}}{1 + e^{-0.1(V_m+32)}} & V_m \geq -40\mathrm{mV} \end{cases} \tag{A.1.4c}$$

Slow inward current

$$I_{\mathrm{si}} = \overline{G}_{\mathrm{si}} \cdot d \cdot f \cdot (V_m - E_{\mathrm{si}}) \tag{A.1.5}$$

$$E_{\mathrm{si}} = 7.7 - 13.0287 \cdot \ln\left([\mathrm{Ca}]_i\right) \tag{A.1.6}$$

Activation gate, d

$$\frac{dd}{dt} = \alpha_d(1-d) - \beta_d d \tag{A.1.7a}$$

$$\alpha_d = \frac{0.095e^{-0.01(V_m-5)}}{1+e^{-0.072(V_m-5)}} \tag{A.1.7b}$$

$$\beta_d = \frac{0.07e^{-0.017(V_m+44)}}{1+e^{0.05(V_m+44)}} \tag{A.1.7c}$$

Inactivation gate, f

$$\frac{df}{dt} = \alpha_f(1-f) - \beta_f f \tag{A.1.8a}$$

$$\alpha_f = \frac{0.012e^{-0.008(V_m+28)}}{1+e^{0.15(V_m+28)}} \tag{A.1.8b}$$

$$\beta_f = \frac{0.0065e^{-0.02(V_m+30)}}{1+e^{-0.2(V_m+30)}} \tag{A.1.8c}$$

Calcium uptake

$$\frac{d\left([\mathrm{Ca}]_i\right)}{dt} = -10^{-4}I_{\mathrm{si}} + 0.07(10^{-4} - [\mathrm{Ca}]_i) \tag{A.1.9}$$

Outward Currents
Time-dependent potassium current

$$I_{\mathrm{K}} = \overline{G}_{\mathrm{K}} \cdot X \cdot X_{\mathrm{i}} \cdot (V_m - E_{\mathrm{K}}) \tag{A.1.10}$$

$$\overline{G}_{\mathrm{K}} = 0.282 \cdot \sqrt{[\mathrm{K}]_o/5.4} \tag{A.1.11}$$

Activation gate, X

$$\frac{dX}{dt} = \alpha_X(1-X) - \beta_X X \tag{A.1.12a}$$

$$\alpha_X = \frac{0.0005e^{0.083(V_m+50)}}{1+e^{0.057(V_m+50)}} \tag{A.1.12b}$$

$$\beta_X = \frac{0.0013e^{-0.06(V_m+20)}}{1+e^{-0.04(V_m+20)}} \tag{A.1.12c}$$

Inactivation gate, X_{i}

$$X_{\mathrm{i}} = \begin{cases} \dfrac{2.837(e^{0.04(V_m+77)}-1)}{(V_m+77)e^{0.04(V_m+35)}} & V_m > -100\mathrm{mV} \\ 1 & V_m \leq -100\mathrm{mV} \end{cases} \tag{A.1.13}$$

Time-independent potassium current

$$I_{\mathrm{K1}} = \overline{G}_{\mathrm{K1}} \cdot \mathrm{K1}_\infty \cdot (V_m - E_{\mathrm{K1}}) \tag{A.1.14}$$

$$\overline{G}_{\mathrm{K1}} = 0.6047 \cdot \sqrt{[\mathrm{K}]_o/5.4} \tag{A.1.15}$$

Inactivation gate, K1

$$\mathrm{K1}_\infty = \frac{\alpha_{\mathrm{K1}}}{\alpha_{\mathrm{K1}} + \beta_{\mathrm{K1}}} \tag{A.1.16a}$$

$$\alpha_{\mathrm{K1}} = \frac{1.02}{1 + e^{0.2385(V_m - E_{\mathrm{K1}} - 59.215)}} \tag{A.1.16b}$$

$$\beta_{\mathrm{K1}} = \frac{0.49124e^{0.08032(V_m - E_{\mathrm{K1}} + 5.476)} + e^{0.06175(V_m - E_{\mathrm{K1}} - 594.31)}}{1 + e^{-0.5143(V_m - E_{\mathrm{K1}} + 4.753)}} \tag{A.1.16c}$$

Plateau potassium current

$$I_{\mathrm{Kp}} = \overline{G}_{\mathrm{KP}} \cdot \mathrm{Kp} \cdot (V_m - E_{\mathrm{Kp}}) \tag{A.1.17}$$

$$E_{\mathrm{Kp}} = E_{\mathrm{K1}} \tag{A.1.18}$$

$$\mathrm{Kp} = \frac{1}{1 + e^{(7.488 - V_m)/5.98}} \tag{A.1.19}$$

Background potassium current

$$I_{\mathrm{b}} = \overline{G}_{\mathrm{b}} \cdot (V_m - E_{\mathrm{b}}) \tag{A.1.20}$$

Total ionic current

$$\begin{aligned} I_{\mathrm{ion}} &= I_{\mathrm{Na}} + I_{\mathrm{si}} + I_{\mathrm{K}} + I_{\mathrm{K1}} + I_{\mathrm{Kp}} + I_{\mathrm{b}} \\ &= \overline{G}_{\mathrm{Na}} \cdot m^3 \cdot h \cdot j \cdot (V_m - E_{\mathrm{Na}}) + \overline{G}_{\mathrm{si}} \cdot d \cdot f \cdot (V_m - E_{\mathrm{si}}) \\ &\quad + \overline{G}_{\mathrm{K}} \cdot X \cdot X_{\mathrm{i}} \cdot (V_m - E_{\mathrm{K}}) + \overline{G}_{\mathrm{K1}} \cdot \mathrm{K1}_{\mathrm{inf}} \cdot (V_m - E_{\mathrm{K1}}) \\ &\quad + \overline{G}_{\mathrm{KP}} \cdot \mathrm{Kp} \cdot (V_m - E_{\mathrm{Kp}}) + \overline{G}_{\mathrm{b}} \cdot (V_m - E_{\mathrm{b}}) \end{aligned} \tag{A.1.21}$$

For an individual cardiac cell we have that the transmembrane potential V_m is given by [34]:

$$\frac{dV_m}{dt} = -\frac{1}{C_m}(I_{ion} + I_{st}), \tag{A.1.22}$$

where C_m is the membrane capacitance and I_{st} is the stimulus current applied by the sinoatrial node.

The following table shows the values of the channel conductances, the reversal potentials for the ions, and other parameters.

Table A.1: Parameters for the Luo-Rudy Phase I model; the conductances are in $\mathrm{mS/cm^2}$ and the reversal potentials in mV [9].

Channel Conductance		Reversal Potential		Other Parameters
$\overline{G}_{\mathrm{Na}} = 23.0$		$E_{Na} = 54.4$		Resting Membrane Potential $V_{rest} = -84.0$mV
$\overline{G}_{\mathrm{si}} = 0.09$		$E_{si} = 118.7$		Membrane Threshold Potential $V_{threshold} = -60$mV
$\overline{G}_{\mathrm{K}} = 0.282$		$E_K = -77$		$[K]_o = 5.4$mM
$\overline{G}_{\mathrm{K1}} = 0.6047$		$E_{K1} = -87.2$		Membrane Capacitance $C_m = 1\ \mu\mathrm{F/cm^2}$
$\overline{G}_{\mathrm{Kp}} = 0.0183$		$E_{Kp} = -87.2$		
$\overline{G}_{\mathrm{b}} = 0.03921$		$E_b = -59.87$		

A.2 The model of Courtemanche et al.

The transmembrane potential, V_m, is given by

$$\frac{dV_m}{dt} = -\frac{1}{C_{\rm m}}(I_{\rm ion} + I_{\rm st}),$$

where $I_{\rm ion}$ is defined as

$$\begin{aligned} I_{\rm ion} &= I_{\rm Na} + I_{\rm K1} + I_{\rm to} + I_{\rm Kur} + I_{\rm Kr} + I_{\rm Ks} \\ &+ I_{\rm Ca,L} + I_{\rm p,Ca} + I_{\rm NaK} + I_{\rm NaCa} + I_{\rm b,Na} + I_{\rm b,Ca}, \end{aligned}$$

and $I_{\rm st}$ is the stimulus current. There are 15 gating equations in the form

$$\frac{dy}{dt} = \frac{y_\infty - y}{\tau_y}, \tag{A.2.1}$$

where y is the gating variable and $y_{\rm inf}$ and τ_y are defined as

$$\begin{aligned} y_\infty &= \frac{\alpha_y}{\alpha_y + \beta_y}, \\ \tau_y &= \frac{1}{\alpha_y + \beta_y}, \end{aligned}$$

with both α_y and β_y being functions of V. The remaining ODEs relate to ionic concentrations and are defined as

$$\begin{aligned} \frac{d[{\rm Na}^+]_{\rm i}}{dt} &= \frac{-3I_{\rm Na,K} - 3I_{\rm NaCa} - I_{\rm b,Na} - I_{\rm Na}}{F{\rm V_i}}, \\ \frac{d[{\rm K}^+]_{\rm i}}{dt} &= \frac{2I_{\rm Na,K} - I_{\rm K1} - I_{\rm to} - I_{\rm Kur} - I_{\rm Kr} - I_{\rm Ks} - I_{\rm b,K}}{F{\rm V_i}}, \\ \frac{d[{\rm Ca}^{2+}]_{\rm i}}{dt} &= \frac{B1}{B2}, \\ B1 &= \frac{2I_{\rm NaCa} - I_{\rm p,Ca} - I_{\rm Ca,L} - I_{\rm b,Ca}}{2F{\rm V_i}} \\ &+ \frac{{\rm V_{up}}(I_{\rm up,leak} - I_{\rm up}) + I_{\rm rel}{\rm V_{rel}}}{\rm V_i}, \\ B2 &= 1 + \frac{[{\rm Trpn}]_{\rm max}K_{\rm m,Trpn}}{([{\rm Ca}_{\rm i}^{2+}] + K_{\rm m,Trpn})^2} + \frac{[{\rm Cmdn}]_{\rm max}K_{\rm m,Cmdn}}{([{\rm Ca}^{2+}]_{\rm i} + K_{\rm m,Cmdn})^2}, \\ \frac{d[{\rm Ca}^{2+}]_{\rm up}}{dt} &= I_{\rm up} - I_{\rm up,leak} - I_{\rm tr}\frac{\rm V_{rel}}{\rm V_{up}}, \\ \frac{d[{\rm Ca}^{2+}]_{\rm rel}}{dt} &= (I_{\rm tr} - I_{\rm rel})\Big\{1 + \frac{[{\rm Csqn}]_{\rm max}K_{\rm m,Csqn}}{([{\rm Ca}^{2+}]_{\rm rel} + K_{\rm m,Csqn})^2}\Big\}^{-1}. \end{aligned}$$

For further details, see [11].

A.3 The model of Winslow et al.

The transmembrane potential, V_m, is defined as

$$\begin{aligned} \frac{dV_m}{dt} &= -(I_{\rm Na} + I_{\rm Ca} + I_{\rm Ca,K} + I_{\rm Kr} + I_{\rm Ks} + I_{\rm to1} + I_{\rm K1} + I_{\rm Kp} \\ &+ I_{\rm NaCa} + I_{\rm NaK} + I_{\rm p(Ca)} + I_{\rm Ca,b} + I_{\rm Na,b}). \end{aligned}$$

There are eight gating equations to describe sodium and potassium:

$$\begin{aligned}
\frac{dm}{dt} &= \alpha_m(1-m) - \beta_m m, \\
\frac{dh}{dt} &= \alpha_h(1-h) - \beta_h h, \\
\frac{dj}{dt} &= \alpha_j(1-j) - \beta_j j, \\
\frac{dX_{\mathrm{Kr}}}{dt} &= K_{12}(1-X_{\mathrm{Kr}}) - K_{21}X_{\mathrm{Kr}}), \\
\frac{dX_{\mathrm{Ks}}}{dt} &= \frac{(X_{\mathrm{Ks}}^{\infty} - X_{\mathrm{Ks}})}{\tau_{X_{\mathrm{Ks}}}}, \\
\frac{dX_{\mathrm{to1}}}{dt} &= \alpha_{X_{\mathrm{to1}}}(1-X_{\mathrm{to1}}) - \beta_{X_{\mathrm{to1}}}X_{\mathrm{to1}}, \\
\frac{dY_{\mathrm{to1}}}{dt} &= \alpha_{Y_{\mathrm{to1}}}(1-Y_{\mathrm{to1}}) - \beta_{Y_{\mathrm{to1}}}Y_{\mathrm{to1}}, \\
\frac{dy}{dt} &= \frac{y_{\infty} - y}{\tau_y}.
\end{aligned}$$

There are a number of equations related to calcuim concentration:

$$\frac{dP_{C_1}}{dt} = -k_a^+[\mathrm{Ca}^{2+}]_{\mathrm{ss}}^n P_{C_1} + k_a^- P_{O_1}, \tag{A.3.1}$$

$$\begin{aligned}
\frac{dP_{O_1}}{dt} &= k_a^+[\mathrm{Ca}^{2+}]_{\mathrm{ss}}^n P_{C_1} - k_a^- P_{O_1}, -k_b^+[\mathrm{Ca}^{2+}]_{\mathrm{ss}}^m P_{O_1} \\
&\quad + k_b^- P_{O_2} - k_c^+ P_{O_1} + k_c^- P_{C_2},
\end{aligned} \tag{A.3.2}$$

$$\frac{dP_{O2}}{dt} = k_b^+[\mathrm{Ca}^{2+}]_{\mathrm{ss}}^m P_{O_1} - k_b^- P_{O_2}, \tag{A.3.3}$$

$$\frac{dP_{C_2}}{dt} = k_c^+ P_{O_1} - k_c^- P_{C_2}. \tag{A.3.4}$$

The following system describes the membrane current of calcium through the so-called L-type channels.

$$\begin{aligned}
\frac{dC_0}{dt} &= \beta C_1 + \omega C_{\mathrm{Ca0}} - (4\alpha + \gamma)C_0, \\
\frac{dC_1}{dt} &= 4\alpha C_0 + 2\beta C_2 + \frac{\omega}{b}C_{\mathrm{Ca1}} - (\beta + 3\alpha + \gamma a)C_1, \\
\frac{dC_2}{dt} &= 3\alpha C_1 + 3\beta C_3 + \frac{\omega}{b^2}C_{\mathrm{Ca2}} - (2\beta + 2\alpha + \gamma a^2)C_2, \\
\frac{dC_3}{dt} &= 2\alpha C_2 + 4\beta C_4 + \frac{\omega}{b^3}C_{\mathrm{Ca3}} - (3\beta + \alpha + \gamma a^3)C_3, \\
\frac{dC4}{dt} &= \alpha C_3 + gO + \frac{\omega}{b^4}C_{\mathrm{Ca4}} - (4\beta + f + \gamma a^4)C_4, \\
\frac{dO}{dt} &= fC_4 - gO, \\
\frac{dC_{\mathrm{Ca0}}}{dt} &= \beta' C_{\mathrm{Ca1}} + \gamma C_0 - (4\alpha' + \omega)C_{\mathrm{Ca0}},
\end{aligned}$$

$$
\begin{aligned}
\frac{dC_{\mathrm{Ca1}}}{dt} &= 4\alpha' C_{\mathrm{Ca0}} + 2\beta' C_{\mathrm{Ca2}} + \gamma a C_1 - (\beta' + 3\alpha' + \frac{\omega}{b})C_{\mathrm{Ca1}},\\
\frac{dC_{\mathrm{Ca2}}}{dt} &= 3\alpha' C_{\mathrm{Ca1}} + 3\beta' C_{\mathrm{Ca3}} + \gamma a^2 C_2 - (2\beta' + 2\alpha' + \frac{\omega}{b^2})C_{\mathrm{Ca2}},\\
\frac{dC_{\mathrm{Ca3}}}{dt} &= 2\alpha' C_{\mathrm{Ca2}} + 4\beta' C_{\mathrm{Ca4}} + \gamma a^3 C_3 - (3\beta' + \alpha' + \frac{\omega}{b^3})C_{\mathrm{Ca3}},\\
\frac{dC_{\mathrm{Ca4}}}{dt} &= \alpha' C_{\mathrm{Ca3}} + \gamma a^4 C_4 - (4\beta' + f' + \frac{\omega}{b^4})C_{\mathrm{Ca4}},
\end{aligned}
$$

Intracellular calcium buffering is described by

$$
\begin{aligned}
\frac{d[\mathrm{HTRPNCa}]}{dt} &= k^+_{\mathrm{htrpn}}[\mathrm{Ca}^{2+}]_i([\mathrm{HTRPN}]_{\mathrm{tot}} - [\mathrm{HTRPNCa}])\\
&\quad -k^-_{\mathrm{htrpn}}[\mathrm{HTRPNCa}],\\
\frac{d[\mathrm{LTRPNCa}]}{dt} &= k^+_{\mathrm{ltrpn}}[\mathrm{Ca}^{2+}]_i([\mathrm{LTRPN}]_{\mathrm{tot}} - [\mathrm{LTRPNCa}])\\
&\quad -k^-_{\mathrm{ltrpn}}[\mathrm{LTRPNCa}],
\end{aligned}
$$

where the k-coefficients are constants.

Intracellular ionic concentrations are described by:

$$
\begin{aligned}
\frac{d[\mathrm{Na}^+]_i}{dt} &= -(I_{Na} + I_{Na,b} + 3I_{NaCa} + 3I_{NaK})fracA_{cap}C_{sc}V_{myo}F,\\
\frac{d[\mathrm{K}^+]_i}{dt} &= -(I_{\mathrm{Kr}} + I_{\mathrm{Ks}} + I_{\mathrm{to1}} + I_{\mathrm{K1}},\\
&\quad +I_{\mathrm{Kp}} + I_{\mathrm{Ca,K}} - 2I_{\mathrm{NaK}})\frac{A_{\mathrm{cap}}C_{\mathrm{sc}}}{V_{\mathrm{myo}}F},\\
\frac{d[\mathrm{Ca}^{2+}]_i}{dt} &= \beta_i\Big[J_{\mathrm{xfer}} - J_{\mathrm{up}} - J_{\mathrm{trpn}}\\
&\quad -(I_{\mathrm{Ca,b}} - 2I_{\mathrm{NaCa}} + I_{\mathrm{p(Ca)}})\frac{A_{\mathrm{cap}}C_{\mathrm{sc}}}{2V_{\mathrm{myo}}F}\Big],\\
\frac{d[\mathrm{Ca}^{2+}]_{\mathrm{ss}}}{dt} &= \beta_{\mathrm{ss}}\left(J_{\mathrm{rel}}\frac{V_{\mathrm{JSR}}}{V_{\mathrm{myo}}} - J_{\mathrm{xfer}}\frac{V_{\mathrm{myo}}}{V_{\mathrm{ss}}} - I_{\mathrm{Ca}}\frac{A_{\mathrm{cap}}C_{\mathrm{sc}}}{2V_{\mathrm{myo}}F}\right),\\
\frac{d[\mathrm{Ca}^{2+}]_{\mathrm{JSR}}}{dt} &= \beta_{\mathrm{JSR}}(J_{\mathrm{tr}} - J_{\mathrm{rel}}),\\
\frac{d[\mathrm{Ca}^{2+}]_{\mathrm{NSR}}}{dt} &= J_{\mathrm{up}}\frac{V_{\mathrm{myo}}}{V_{\mathrm{NSR}}} - J_{\mathrm{tr}}\frac{V_{\mathrm{JSR}}}{V_{\mathrm{NSR}}}.
\end{aligned}
$$

There are 33 ODEs in total. See [69] for details.

A.4 The Puglisi–Bers model

The transmembrane potential, V_m, is given by

$$
\begin{aligned}
\frac{dV_m}{dt} &= \frac{I_{stim} - (I_{Na} + I_{Ca_L} + I_{Ca_T} + I_{Kr} + I_{Ks} + I_{NaCa} + I_{K1} + I_{Kp})}{C}\\
&+ \frac{I_{stim} - (I_{p_{Ca}} + I_{Na_b} + I_{Ca_b} + I_{NaK} + I_{to} + I_{Cl(Ca)})}{C}
\end{aligned}
$$

There are nine gating equations:

$$\begin{aligned}
\frac{dm}{dt} &= \alpha_m(1-m) - \beta_m m, \\
\frac{dh}{dt} &= \alpha_h(1-h) - \beta_h h, \\
\frac{dj}{dt} &= \alpha_j(1-j) - \beta_j j, \\
\frac{dd}{dt} &= \alpha_d(1-d) - \beta_d d, \\
\frac{df}{dt} &= \alpha_f(1-f) - \beta_f f, \\
\frac{db}{dt} &= \frac{b_\infty - b}{\tau_b}, \\
\frac{dg}{dt} &= \frac{g_\infty - g}{\tau_g}, \\
\frac{dX_r}{dt} &= \frac{X_{r_\infty} - X_r}{\tau_{X_r}}, \\
\frac{dX_s}{dt} &= \frac{X_{s_\infty} - X_s}{\tau_{X_s}}
\end{aligned}$$

There are seven equations to describe ionic concentrations:

$$\begin{aligned}
\frac{dNa_i}{dt} &= -(I_{Na} + I_{CaNa} + I_{Na_b} + 3I_{NaCa} + 3I_{NaK})\frac{A_{cap}}{V_{myo}F}, \\
\frac{dCa_i}{dt} &= ((I_{CaCa} + I_{p_{Ca}} + I_{Ca_b} + I_{Ca_T}) - I_{NaCa})\frac{A_{cap}}{2V_{myo}F} + I_{rel}\frac{V_{JSR}}{V_{myo}} + (I_{leak} - I_{up})\frac{V_{NSR}}{V_{myo}}, \\
\frac{dK_i}{dt} &= -(I_{CaK} + I_{Kr} + I_{Ks} + I_{K1} + I_{Kp} + I_{to} - 2I_{NaK})\frac{A_{cap}}{V_{myo}F}, \\
\frac{dK_o}{dt} &= (I_{CaK} + I_{Kr} + I_{Ks} + I_{K1} + I_{Kp} + I_{to} - 2I_{NaK})\frac{A_{cap}}{V_{cleft}F}, \\
\frac{dCa_{JSR}}{dt} &= -(I_{rel} - I_{tr}\frac{V_{NSR}}{V_{JSR}}), \\
\frac{dCa_{NSR}}{dt} &= -(((I_{leak} + I_{tr}) - I_{up})), \\
\frac{dCa_{foot}}{dt} &= (I_{CaCa})\frac{A_{cap}}{2V_{myo}F}R_{A_V}
\end{aligned}$$

There are 17 ODEs in total. See [45] and the references within for more details.