8-2011

A TIME-AND-SPACE PARALLELIZED ALGORITHM FOR THE CABLE EQUATION

Chuan Li
cli9@utk.edu

To the Graduate Council:

I am submitting herewith a dissertation written by Chuan Li entitled "A TIME-AND-SPACE PARALLELIZED ALGORITHM FOR THE CABLE EQUATION." I have examined the final electronic copy of this dissertation for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Doctor of Philosophy, with a major in Mathematics.

Vasilios Alexiades, Major Professor

We have read this dissertation and recommend its acceptance:

Ohannes Karakashian, Xiaobing Feng, Jack Buchanan

Accepted for the Council:
Carolyn R. Hodges

Vice Provost and Dean of the Graduate School

(Original signatures are on file with official student records.)

A TIME-AND-SPACE PARALLELIZED ALGORITHM FOR THE CABLE EQUATION

A Dissertation

Presented for the

Doctor of Philosophy

Degree

The University of Tennessee, Knoxville

Chuan Li

August 2011

This dissertation is dedicated to all of people who have helped and guided me throughout my research, including my committee advisors, Vasilios Alexiades, Ohannes Karakashian, Xiaobing Feng, Jack Buchanan, and many other professors who have provided me with an excellent level of instruction throughout all of my course work here at the University of Tennessee, Knoxville. Also, I would like to dedicate this work to my parents, my wife and my son, for their understanding of my situation as a graduate student and for their support throughout this endeavor.

Acknowledgements

I would like to express my gratitude to my committee members, who have taken time to review my work and to pose questions which have allowed me to grow as an academic throughout this process. A special acknowledgment is in order for my advisor, Prof. Vasilios Alexiades, who took a chance in teaching me how to become a researcher.

Abstract

Electrical propagation in excitable tissue, such as nerve fibers and heart muscle, is described by a nonlinear diffusion-reaction parabolic partial differential equation for the transmembrane voltage $V(x,t)$, known as the *cable equation*. This equation involves a highly nonlinear source term, representing the total ionic current across the membrane, governed by a Hodgkin-Huxley type ionic model, and requires the solution of a system of ordinary differential equations. Thus, the model consists of a PDE (in 1-, 2- or 3-dimensions) coupled to a system of ODEs, and it is very expensive to solve, especially in 2 and 3 dimensions.

To solve this problem numerically, we develop and implement an extension of the time-parallel Parareal Algorithm, introduced by Lions-Maday-Turinici in 2001, to efficiently incorporate space-parallelized solvers into the time-parallelization framework of the Parareal algorithm, to achieve **time-and-space parallelization**.

We analyze the speedup, efficiency, and scaling of space-only, time-only, and time-and-space parallel algorithms, and determine conditions under which each is likely to perform well.

We present numerical results and comparison of the performance of several serial, space-parallelized and time-and-space-parallelized time-stepping numerical schemes in one-dimension and in two-dimensions on the electrical potential propagation problem.

Finally, we conduct extensive numerical experiments of action potential propagation in cardiac tissue in one and two dimensions, to determine the effect of varying

certain biological parameters and their influence on the action potential duration and propagation velocity.

Contents

List of Tables

List of Figures

Chapter 1

BIOLOGICAL INTRODUCTION

The heart is considered to be the most important organ, and also most studied, in the human body for thousands of years. Figure 1.1 shows the structure of the human heart.
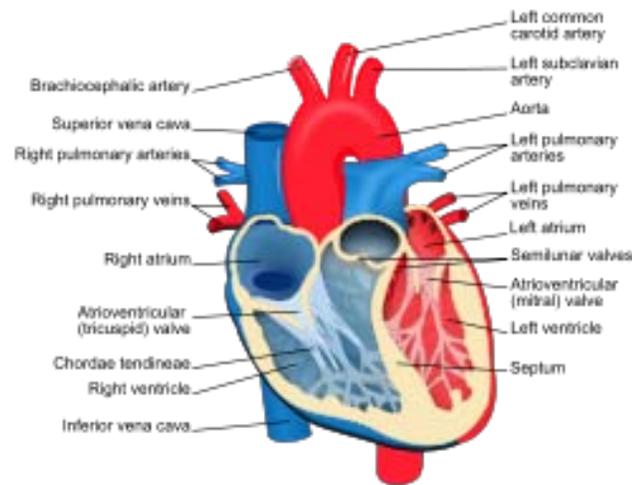
Figure 1.1: Structure diagram of the human heart from an anterior view [46].

In the present era, cardiology clinics and research use an enormous number of cutting-edge computer techniques. One such technique employs computers in medical research and constructs mathematical and computer models of biomedical systems

and processes. In several circumstances the use of these models has now reached the stage of practical clinical usefulness.

An iterative process of conducting cardiac studies is described in [53] as follows. First, a mathematical model/equation for certain biological/physical quantities is established. Next, numerical techniques are employed to solve the equation and an approximate solution at a number of discrete grids is obtained. Finally, the results have to be validated by comparing them to physical measurements. For a complex phenomenon such as the electrical activity of the heart, it is likely that significant differences will be observed between measured and simulated values. The differences are usually caused by limitations in the mathematical model. In this case, a refinement of the mathematical model is required, and previous steps will be repeated on the refined model. This process is illustrated in Figure 1.2.
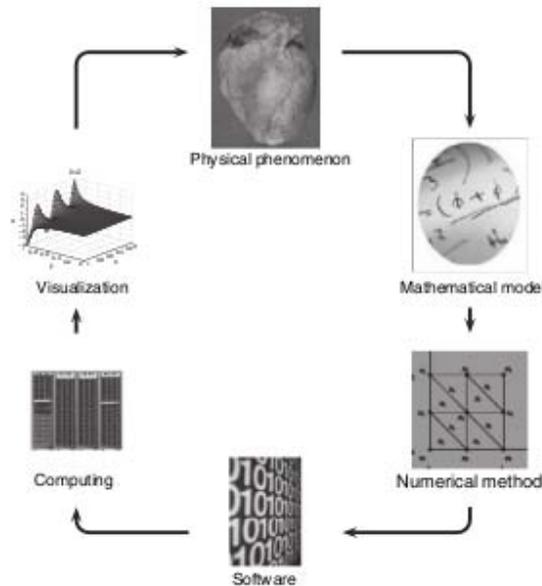
Figure 1.2: The mathematical models are developed through a continuous interaction between modelling, computer simulations, and physical experiments [53].

According to [39, 40, 41], mathematical and computer models aiming at reproducing cardiac electrophysiological processes can be classified into four categories,

depending on the level of detail at which the processes are simulated. (1) Models in the most detailed level simulate intracellular processes, such as membrane ionic currents, of individual cells. (2) Models/Systems in a less accurate level simulate blocks of cardiac tissue composed of several tens or hundreds of cells and reproduce the processes of the interactions between neighboring cells and neighboring blocks of cells. (3) Models in an even lower level of detail study excitation properties and cardiac electric fields related either to the entire heart or to entire ventricles (including their geometrical properties). (4) Marcroexcitation models in the lowest level of detail divide the entire heart into a small number of regions (such as the right and left atrium, atrioventricular node, left and right ventricle, etc.) and treat each of these parts as an elementary unit [39, 40]. Global excitation properties and arrhythmia manifestation can be modeled in this way [41]. This classification is shown in Figure 1.3.
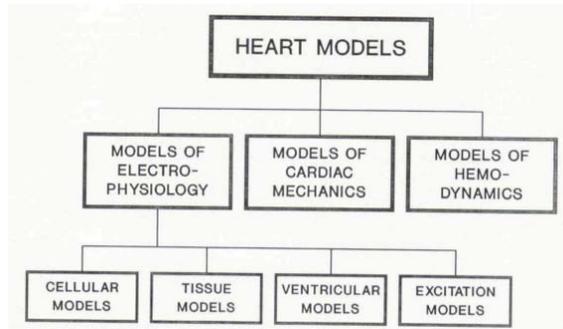
Figure 1.3: Schematic classification of heart and mathematical & computer models [40].

This research focuses on electrophysiological models of individual cells and of bundles of cells, that simulate intracellular processes such as membrane ionic currents and propagation of action potentials. A brief history of developments and studies of cardiac cellular models is presented in the next section.

1.1 Overview of Cardiac Cellular Models

In the early 1950's, Hodgkin and Huxley analysed ionic currents during the excitation of squid nerve axon and constructed mathematical equations modeling the action potential and activation conduction, based on changes of concentrations of sodium and potassium and their transmembrane conductances [25]. They won the Nobel Prize in Physiology for their pioneering work in 1963. The Hodgkin-Huxley model is the first successful mathematical model describing the behavior of excitable cells. Moreover, the basic excitation conduction mechanism may be applicable to other excitable cells, though very different parameters might be required to account for the large differences in action potentials in different cells.

Since the Hodgkin-Huxley model was published, various modifications and improvements of the mathematical model have been developed to make it more appropriate for cardiac muscle cells of different species. These modifications and improvements are based on recent experiments and observations to remove the restrictive assumption of constant ionic concentrations in the Hodgkin-Huxley model, and incorporate the sodium pump currents, sodium-calcium currrent exchangers and the presence of several types of calcium channels with varied kinetics into the modeling equations [40] (Figure 1.4). With more and more details about the ionic activity of cellular membranes and intracellular particles are incorporated, most recent models are constructed to reproduce the activity of individual cell types due to the differences between types of cardiac cells are significant [40].

The Luo-Rudy Phase I model [32] is one of these improved models, which will be used in all our simulations. It will be described in detail in §2.2.2.

Recent simulation studies use these models to investigate ionic potentials and currents as well as other aspects of cardiac cell behavior. An example, described in [40], studies the infulence of heart rate on ionic concentrations and simulates cellular mechanisms of excitation-contraction coupling. The results are shown in Figure 1.5.
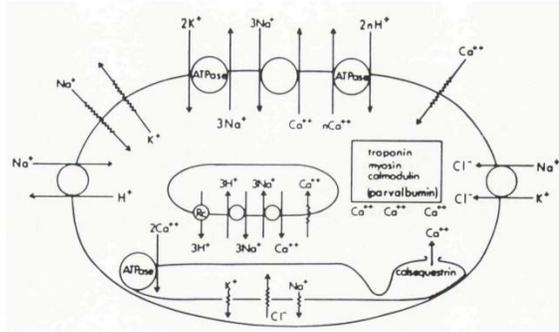
Figure 1.4: Basic structure of the ionic regulation system for sodium, potassium, and calcium in myocytes. This represents only a minimum set of transport mechanisms incorporated into a model of ionic currents. The large oval represents the surface membrane of a single cell: the central oval represents the mitochondrial membrane; and the compartment at the bottom of the picture images the sarcoplasmic reticulum. The authors of the model point out that the cell nucleus and intranuclear calcium transport should also be incorporated. Within the model, each transport mechanism must be described by an individual equation [40].

Figure 1.5: An example of the results of a cellular modeling study simulating rabbit atrial myocytes during regular stimulation. Curve i shows the modeled action potential; curve ii demonstrates the total cytosolic calcium (left calibration axis); curve iii shows contraction, expressed as a percentage of maximum contraction(right calibration axis); and curve iv expresses the extracellular calcium transient. The inset of the figure in the right upper corner shows experimental data corresponding to curve i, iii, and iv. Note the close resemblance between these experimental observations and the simulated results [40].
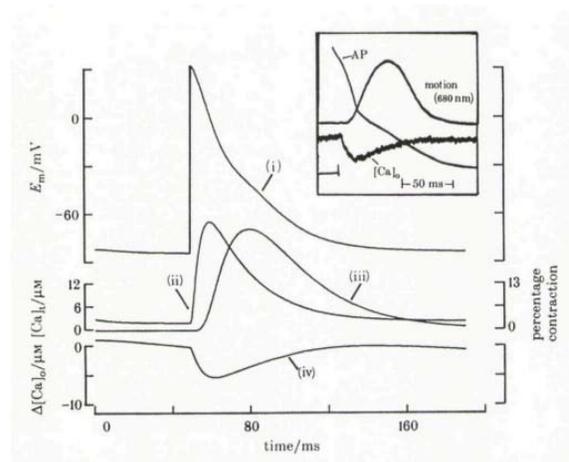
One important subject in cardiac simulation studies is modeling of cardiac arrhythmias, pathological conditions under which regular activation may decay into complex and irregular patterns that impair functionality of the heart and may lead to death. One goal of my research is to explore if it is possible to induce arrhythmia in one and/ or two-dimensional simulations of cardiac tissue.

1.2 Action Potential and Related Biological Quantities

Action potentials (AP) are voltage waveforms that propagate along a cell and between cells. They are due to selective permeability of ion channels on the plasma membrane of excitable cells.

A variety of action potential types exist in many cell types according to the types of voltage-gated channels, leak channels, channel distributions, ionic concentrations, membrane capacitance, temperature, and other factors.

Neuronal and cardiac APs are often referred to as **"all-or-nothing"**. That is, the action potential either occurs fully (cell is depolarized) or it does not ocurr at all (cell is polarized, near resting state).

This unique all-or-nothing property distinguishes the action potential from graded potentials such as receptor potentials, electrotonic potentials, and synaptic potentials, which scale with the magnitude of the stimulus [57].

When it occurs, the action potential has a characteristic shape, largely independent of the stimulus strength. A typical myocardial action potential is shown in Fig.1.6 with the various phases marked by numbers. The sharply rising phase ("0" in 1.6) is caused by the opening of a large number of Na^+-selective ion channels in response to sufficiently large increase in membrane potential, resulting in influx of sodium and depolarizing the cell. Sodium channels close shortly after they open, causing the small drop ("1"). The falling phase ("3") is due to potassium channels

6

opening in response to depolarization, allowing K^+ ions to flow out of the cell. The characteristic plateau ("2") results from the opening of voltage-sensitive calcium channels.
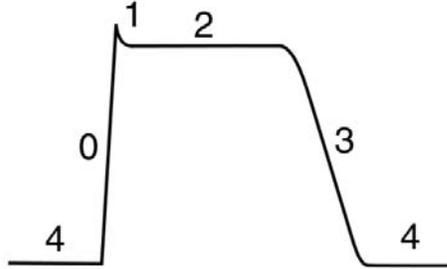
Figure 1.6: Schematic shape of myocardial action potential [57].

Another important property of action potentials is **refractoriness**, which means that it is impossible to evoke a second AP during and for a short time period after an action potential has occurred due to deactivation of sodium channels, which also prevents the AP from propagating backwards. This period is referred to as the *absolute refractory period (ARP)*. An action potential can be evoked, but only by a larger stimulus than was required to evoke the first action potential, in a period referred to as the *relative refractory period (RRP)* after the absolute refractory period. Once transients have settled out, stimulation by an ongoing supra-threshold stimulus leads to repetitive constant firing [56].

In cardiac myocytes, the action potential travels smoothly in the cell, with constant shape and at constant velocity once initiated. The leading edge of the action potential depolarizes adjacent unexcited portions of the cell and brings them to threshold. In the wake of the action potential, the membrane is refractory and prevents re-exciting previously active portions of the cell [56].

Certain quantities pertaining to an action potential are characteristic of the cell and the ionic currents, and independent of the stimulus, cable length, and locations used for measurement. Thus these quantities also serve as accuracy indicators on the numerical schemes. Such important quantites, which are tracked in our simulations, are the following:

7

- **Action Potential Duration (APD):** The duration is determined by measuring how long the potential V at a specific location stays above a certain cut-off value. In our computations, APD is determined by setting a cut-off voltage at 90% of the initial equilibrium voltage.

  We record APDs at two locations, one near the left end and one near the right end, denoted respectively as $APD0$ and $APD1$ in the tables later. These locations are user-settable in the codes; usually $APD0$ is measured at node 100 and $APD1$ at node $(M - 100)$, with M the total number of control volumes.

- **Propagation Velocity:** It measures how fast the action potential propagates along the cable. It is measured by the difference of the starting time of APs at two specified locations.

- **Maximum voltage (V_{max}) and maximum rate of change (dV/dt_{max}).**

1.3   Arrhythmia

Cardiac arrhythmia is a dangerous heart condition that may lead to suddent death. According to [1], each year about 295,000 emergency medical services-treated out-of-hospital cardiac arrests occur in the United States, and many of them are due to cardiac arrhythmias. Arrhythmias occur throughout the population and may result from either *ventricular tachycardia (VT)/ventricular fibrillation (VF)*, an extremely fast, chaotic rhythm; or *ventricular bradycardia (VB)*, an extremely slow, chaotic rhythm, during which the lower chambers quiver and the heart cannot pump any blood, causing cardiac arrest. Arrhythmias can also cause serious injury to other organs. The brain, kidneys, lungs or liver may be damaged during prolonged cardiac arrest.

The most dangerous cardiac arrhythmias are usually associated with abnormal wave propagation caused by reentrant sources of excitation. A lot of research has been performed on animal hearts (rat, guinea pig, rabbit, dog, etc.), but the mechanisms

underlying the initiation and subsequent dynamics of these reentrant sources in the human heart still remain unknown, not only due to the limited possibilities of invasively studying cardiac arrhythmias in humans [54], but also due to some major limitations of experimental studies of ventricular arrhythmias. One major limitation, for instance, is that the underlying excitation patterns are three dimensional which cannot be recorded only from the surface of the heart in biological laboratory experiments [54]. These limitations make computational modeling, especially detailed quantitative modeling of the human heart, an ideal research approach in cardiology.

Establishing a suitable model describing electrodynamical properties of cardiac cell bundles is the core of any cardiac arrhythmia modeling study. Most detailed electrophysiological models have been formulated for animal cardiomyocytes because of the limitations of experimental research described above. For example, the Noble model (Noble et al., 1998) and the Luo-Rudy models (Luo and Rudy, 1991, 1994) were formulated for guinea pig ventricular cells, whereas the Winslow (Winslow et al., 1999) model was formulated for canine ventricular cells [54].

Though models for animal cardiomyocytes are useful, models for human ventricular myocytes are still much needed since animal cardiomyocytes differ from human ones in many important biological aspects (action potential shape and duration, range of normal heart rates, action potential restitution and relative importance of ionic currents in the action potential generation, etc.), and these factors may have noticeable influence on the mechanism of arrhythmia initiation and dynamics [54].

Several models of human ionic currents have benefited from newly developed experimental techniques and been developed for human cardiomyocytes in recent years. Some important ones, mentioned in [54] as well, are the following.

- **PB model:** The first model for human ventricular myocytes published by Priebe and Beuckelmann in 1998 [43].

  This model composed of 15 varibles and was largely based on the Luo-Rudy phase II model for guinea pig ventricular cells [33].

- **redPB model:** A reduced version of the PB model proposed by Bernus, Verschelde and Panfilov in 2002 [7].

  This model reduced to 6 varibles by reformulating some currents and fixating intracellular ionic concentrations.

- **TNNP model:** A new model for human ventricular myocytes introduced by Tusscher, Noble, Noble and Panfilov in 2004 [55].

  This model was constructed as a compromise between physiological detail and computational cost of large-scale spatial simulations. It consisted of 16 varibles and used experimental data from human ventricular cell and ion channel expression experiments to formulate all major ionic currents.

- **IMW model:** Another model for human ventricular myocytes constructed by Iyer, Mazhari and Winslow in 2004 [26].

  This model involved a large number (67) of varibles resulting from formulations for major ionic currents by Markov chain instead of Hodgkin-Huxley type. It was obtained from expression data on human cardiac ion channels, comparing to the TNNP model from data on human ventricular cells.

In light of the above, this work will use the Luo Rudy phase I (1991) ionic model [32], which is one of the most widely used ionic models, serving as prototype of other models for human ventricular cells. It contains all essential features of other models, so it is sufficient for our purpose here, which is to develop efficient numerical algorithms for electrophysiological simulation of cardiac tissue. Moreover, it can be replaced by any other ionic model fairly easily in our modular codes.

Simulation of arrhythmias will be attempted by manipulating values of concentrations of ions and other parameters of the ionic model, as will be seen in the following chapters.

Chapter 2

MATHEMATICAL MODELING

Previous modeling studies were based on single columns of uniform length cardiac cells (Figure 2.1) and suggested that propagation was always "discontinuous", i.e. had rather rapid propagation through the cell bodies, with significant delays, due to the increased resistance of the gap junctions connecting the simulated cells.
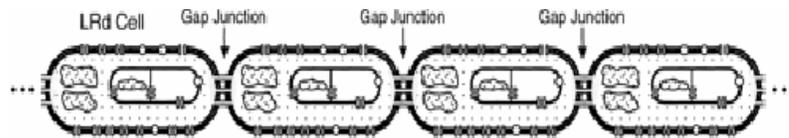
Figure 2.1: A typical single cell column model describing discontinuous propagation [48]

In the mid 1980's, a series of experimental studies using two microelectrodes in line in superfused Guinea pig papillary muscles were conducted by Buchanan et al [12, 11, 19, 20] to improve the general understanding of electrical propagation in the heart by measuring the rate of rise of the upstroke of the action potential, dv/dt_{max}, and the conduction velocity of the action potential simultaneously. Their results, suggesting a square root relationship between dv/dt_{max} and conduction velocity, contradicted the single cell column model and were criticised by others.

Later on, Buchanan et al compared existing models for electrical propagation in the heart to the realities of their experimental preparations, and found several

differences between the models and their preparations. First, single columns of cells almost never occur in vivo. Cell columns aggregate into strands, and strands agrregate into bundles in cardiac tissue. Second, cells are not of uniform length. Legnths of cells randomly distribute between 30 and 130 μm [49]. Third, lateral as well as longitudinal connectivity is provided by abundant physical and functional lateral gap junctions.

In 1990, Buchanan and Fujino constructed new models with the presence of multiple cell columns and the lateral gap junctions. Cell columns in new models were constructed consisting of cells of lenght randomly generated from 30-130 μm by a pseudo-random generator. An element was inserted with intercellular longitudinal resistance 10 times the value of intracellular elements. In addition, lateral gap junctional connections were established between a cell column and its immediate neighbor. Computational results showed that the presence of abundant lateral connectivity led to wavefront smoothing in a two dimensional model and averaged out the "discontinuities" in single cell column one dimensional models [10].

In order to build up a scalable and parallelizable way to model the electrical propagation in the heart and accurately represent membrane ionic currents and the cellular interconnections occurring with a sub-cellular spatial resolution, our codes incorporated the random cell size and gap junctions of the Buchanan and Fujino model in user-settable parameters $minCx$, $maxCx$, $minCy$ and $maxCy$ to describe minimal/maximal cell size in longitudinal and latitudinal directions.

However, it should be pointed out here that all simulation experiments presented in Chapter 5 used fixed $minCx = maxCx = 16\mu m$ for one dimensional experiments, and $minCx = maxCx = 32\mu m$, $minCy = maxCy = 16\mu m$ for two dimensional experiments, only for the sake of eliminating the effects of randomness so that all numerical time-steppers were applied and compared on identical cables in §5.1. Simulations of two dimensional cardiac tissue are described separately, in §5.2.

This chapter is organized as follows: A derivation the cable equation, following [27, 42], is presented in §2.1. Two core ionic models, the Hdgkin-Huxley model and Luo-Rudy phase I (1991) model, are described in §2.2.

2.1 Derivation of the Cable Equation

The cell is assumed to be a cylinder which is isopotential in the radial direction and called a *cable*. Thus, in the electrical description of the cell, the radius is incorporated into the intracellular axial resisitance, resulting in a longitudinally oriented one-dimensional circuit (Figure 2.2) when the cable is divided into a number of segments of length dx of isopotential membrane.
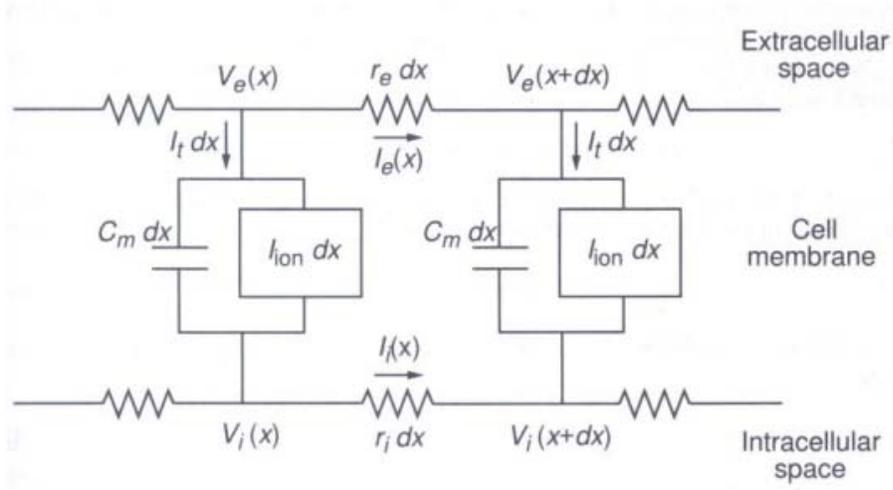
Figure 2.2: Schematic diagram of a discretized cable, with isopotential circuit elements of length dx [27]

The membrane capacitance is C_m, I_i and I_e are the interacellular and extracelluar axial currents, r_i and r_e are the resistances per unit length of the intracellular and extracellular media, respectively.

The *core conductor assumption* [44] states that the potential depends only on the length variable and not on radial or angular variables everywhere along its length. It was mentioned in [27] that two types of current, axial current and transmembrane current (Figure 2.2) must be taken into consideration and balanced in any piece when the cable is divided into a number of segments of length dx of isopotential membrane.

The axial current has intracellular and extracellular components. Both satisfy Ohm's law,

$$V_i(x + dx) - V_i(x) = -I_i(x)r_i dx, \tag{2.1}$$

13

$$V_e(x + dx) - V_e(x) = -I_e(x)r_e dx. \tag{2.2}$$

In general,

$$r_i = -\frac{R_c}{A_i}, \tag{2.3}$$

where R_c is the *cytoplasmic resitivity*, measured in units of Ohm-length, and A_i is the cross-sectional area of the cable. A similar expression holds for the extracellular space.

The minus sign on the right-hand sides is for the convention that positive current is a flow of positive charges from left to right (in the direction of increaing x). If $V_i(x + dx) > V_i(x)$, then positive charges flow in the direction of decreasing x, giving a negative current.

Sending $dx \to 0$ in (2.1) - (2.2) yields,

$$I_i = -\frac{1}{r_i}\frac{\partial V_i}{\partial x}, \tag{2.4}$$

$$I_e = -\frac{1}{r_e}\frac{\partial V_e}{\partial x}. \tag{2.5}$$

Next, by Kirchhoff's law, any change in extracellular or intracellular axial current must be due to a transmembrane current, and thus

$$I_i(x) - I_i(x + dx) = I_t dx = I_e(x + dx) - I_e(x), \tag{2.6}$$

where I_t is the total transmembrane current (positive outward) per unit length of membrane. Again, sending $dx \to 0$ in (2.6) yields

$$I_t = -\frac{\partial I_i}{\partial x} = \frac{\partial I_e}{\partial x}. \tag{2.7}$$

14

In the case of a cable with no additional current sources, the total axial current is a constant $I_t = I_i + I_e$. Defining the *transmembrane potential* as $V = V_i - V_e$ yields

$$-I_t = \frac{r_i + r_e}{r_i r_e}\frac{\partial V_i}{\partial x} - \frac{1}{r_e}\frac{\partial V}{\partial x}, \tag{2.8}$$

from which it follows that

$$\frac{1}{r_i}\frac{\partial V_i}{\partial x} = \frac{1}{r_i + r_e}\frac{\partial V}{\partial x} - \frac{r_e}{r_i + r_e}I_t. \tag{2.9}$$

Substituting (2.4) and (2.9) into (2.7), we obtain

$$I_t = \frac{\partial}{\partial x}\Big(\frac{1}{r_i + r_e}\frac{\partial V}{\partial x}\Big). \tag{2.10}$$

Finally, the transmembrane current I_t is a sum of the capacitive and ionic currents, and thus

$$I_t = p\Big(C_m\frac{\partial V}{\partial t} + I_{ion}\Big) = \frac{\partial}{\partial x}\Big(\frac{1}{r_i + r_e}\frac{\partial V}{\partial x}\Big), \tag{2.11}$$

where p is the perimeter of the cable. Equation (2.11) is referred to as the *cable equation* in [27], with C_m has units of capacitance per unit area of membrane, and I_{ion} has units of current per unit area of membrane.

If a current I_{stim}, with units of current per unit area, is applied across the membrane (as before, taken positive in the outward direction), then the cable equation becomes

$$I_t = p\Big(C_m\frac{\partial V}{\partial t} + I_{ion} + I_{stim}\Big) = \frac{\partial}{\partial x}\Big(\frac{1}{r_i + r_e}\frac{\partial V}{\partial x}\Big). \tag{2.12}$$

With the assumption that r_i and r_e are independent of x and setting the *axial resistance* $R_a = p(r_i + r_e)$, the cable equation becomes

$$C_m\frac{\partial V}{\partial t} = \frac{1}{R_a}\frac{\partial^2 V}{\partial x^2} - I_{ion}(V) - I_{stim}(t), \tag{2.13}$$

15

where V is the transmembrane voltage, R_a and C_m are the axial resistance and membrane capacitance. I_{ion} represents the total ionic current, and I_{stim} is the applied stimulus current.

A consistent set of units for the quantities appearing in (2.13) are: x in cm, t in ms, V in mV, C_m in $\mu F/cm^2$, R_a in $kOhm$, I_{ion} and I_{stim} in $\mu A/cm^2$.

However, these units are not natural and do not quite conform to the units commonly used in electrophysiology. In input data values, we use the more natural units: x in μm, t in ms, V in mV, Ri in $Ohm\,cm$, and then insert a factor of 10^{-3} to convert Ri to $kOhm\,cm$, which results in units of $kOhm$ for Ra.

Equation (2.13) is a parabolic equation with a nonlinear source I_{ion} which must be specified either by the Hodgkin-Huxley model or other more complicated ionic models, such as the Luo-Rudy (1991) model, to be described below.

2.2   Ionic Models

2.2.1   The Hodgkin-Huxley ionic model

Hodgkin and Huxley developed a system of equations describing the electrical activity of the squid giant axon by viewing a segment of the axon as a simple equivalent electrical circuit [25], as shown in Figure 2.3. The membrane separates the extracellular medium from the cytoplasm of the cell and serves as a capacitor with capacitance C_m in the circuit.

Since the model was estabilished by A.L. Hodgkin and A.F. Huxley in the 1950's, it has been applied to many classes of neurons and to other aspects of neuron physiology, making these equations a fundamental tool for studying mechanisms of neuronal behavior. The Hodgkin-Huxley model was the first complete model successfully describing excitability of a single cell. Almost all other more recent and more complicated ionic models simulating excitable cells in various species are based on the equations of the Hodgkin-Huxley model [24].

16
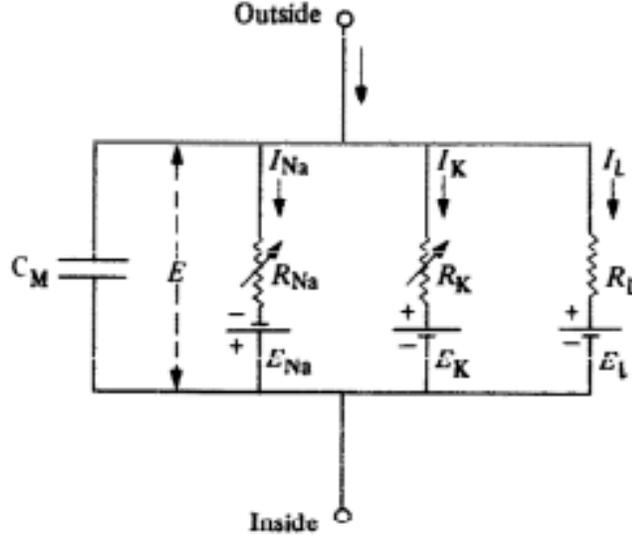
Figure 2.3: Equivalent circuit for Hodgkin-Huxley model of the squid giant axon [25]. $R_{Na} = 1/g_{Na}$, $R_K = 1/g_K$, and $R_L = 1/g_L$. All other quantities are constants.

A brief description of the Hodgkin-Huxley model is given in this section. Detailed mathematical formulation of the model is presented in Appendix A.

In the Hodgkin-Huxley model, the total membrane current I in a excitable cell is a function of time t and membrane voltage V [25]

$$I = C_m \frac{dV}{dt} + I_{ion}(V), \tag{2.14}$$

where C_m is the membrane capacitance and I_{ion} is an applied electrical current. In a propagated action potential, the local circuit currents must be provided by the net membrane current. This fact leads to the relation

$$i = \frac{1}{r_1 + r_2} \frac{\partial^2 V}{\partial x^2}, \tag{2.15}$$

where i is the membrane current per unit length, r_1 and r_2 are the external and internal resistances per unit length, and x is distance along the fiber. r_1 is negligible comparing to r_2 in the case of an axon soaking in a large volume of conducting fluid.

Hence,

$$i = \frac{1}{r_2}\frac{\partial^2 V}{\partial x^2},$$

or equivalently,

$$I = \frac{a}{2R_2}\frac{\partial^2 V}{\partial x^2}, \tag{2.16}$$

where a is the radius of the fiber and R_2 is the specific resistance of the axoplasm. Substituting (2.16) into (2.14) yields an equation of the same form as the cable equation for voltage V,

$$\frac{a}{2R_2}\frac{\partial^2 V}{\partial x^2} = C_m\frac{dV}{dt} + I_{ion}(V). \tag{2.17}$$

The electrical current I_{ion} on the right-hand side of (2.17) consists of three ionic currents: a *sodium current* I_{Na}, a *potassium current* I_K and a *leakage current* I_L,

$$I_{ion}(V) = I_{Na}(V) + I_K(V) + I_L(V). \tag{2.18}$$

These ionic currents depend on three activation and inactivation "gates": m, h and n, which take values between 0 and 1 and are governed by ODEs of the same form

$$\frac{dg}{dt} = \alpha_g(1-g) - \beta_g g, \quad g = m, h, n \tag{2.19}$$

where the α_g's and β_g's are given by explicit formulas as functions of voltage V in [25]. The ionic currents, in turn, change transmembrane voltage V, which subsequently affects the ionic gates and currents.

The Hodgkin-Huxley model is completely described by the system of differential equations (2.17) - (2.19). Hodgkin and Huxley reduced the PDE (2.17) into a 2nd-order ODE (in space) and solved the system numerically during steady propergation, due to the fact that no efficient numerical techniques for solving PDEs was available at that time [25]. At present, with newly developed numerical methods for PDEs, this system can be solved numerically provided the initial values for V, m, h and n at

18

starting time t_0 and boundary condition of V. Figure 2.4 shows how the gates vary in time over 500 ms.
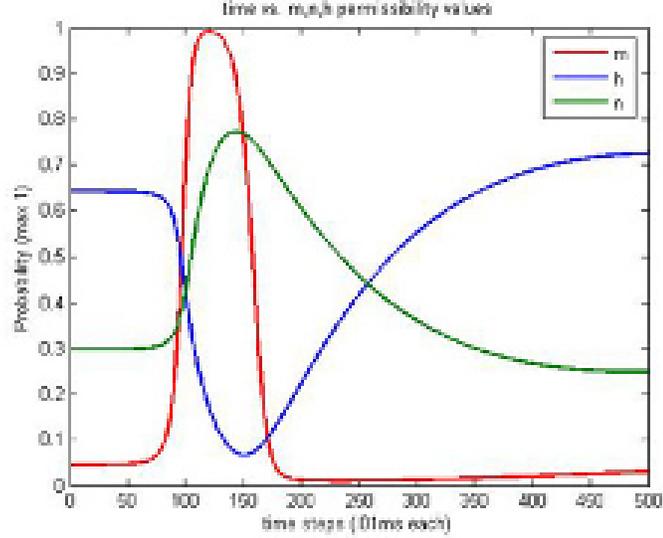
Figure 2.4: Time course of gates of the Hodgkin-Huxley model

2.2.2   The Luo-Rudy Phase I (1991) ionic model

The Luo-Rudy phase I (1991) model in guinea pig ventricular cells was an update of the Beeler-Reuter mammalian ventricular model (1977) (BR model). Like the BR model, the Luo-Rudy phase I (1991) model is an adaptation and extension of the Hodgkin-Huxley formalism to ventricular cells. The Luo-Rudy phase I (1991) model updated the BR model by reformulating the permissive and non-permissive rate coefficients for the sodium current and introducing three new currents: a *time-independent potassium current*, a *plateau potassium current*, and a *time-independent background current*, based on more recent experimental results [32].

The current I_{ion} in this model consists of six ionic currents (while Hudgkin-Huxley has only three):

$$I_{ion}(V) = I_{Na}(V) + I_{SI}(V) + I_K(V) + I_{K1}(V) + I_{Kp}(V) + I_b(V), \qquad (2.20)$$

where I_{Na} is a *fast sodium current*, characterized by fast upstroke velocity and slow recovery from inactivation; I_{SI} is a *slow inward current*; I_K is a *time-dependent potassium current*; I_{K1} is a *time-independent potassium current* that includes a negative-slope phase and displays significant crossover phenomenon as $[K]_o$ is varied; I_{Kp} is a *plateau potassium current*; and I_b is a *time-independent background current*.

The Luo-Rudy phase I (1991) model not only reproduced the effects of $[K]_o$ on action potential duration and rest potential, which will be explored later in §5.1.5 and §5.2, but also demonstated the effects of the slow recovery of I_{Na} in determining the response of the cell [32]. Later on, Luo and Rudy updated their model further to produce the Luo-Rudy Phase II (1994) model [33].

The ionic currents are determined by ionic gates, whose gating variables are obtained as a solution to a coupled system of seven highly nonlinear ODEs, which are of the same form as (2.19). Typical behavior of the gates is shown in Figure 2.5.
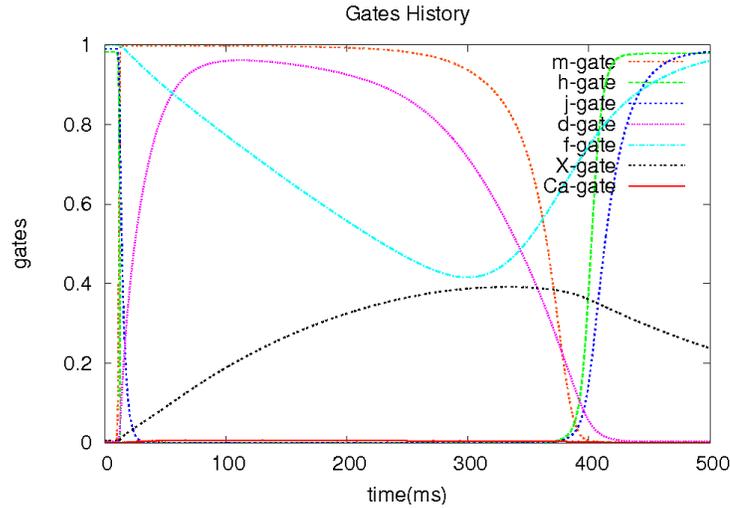
Figure 2.5: Time course of gates of the Luo-Rudy model

The complete mathematical formulation of the Luo-Rudy phase I (1991) model is presented in Appendix B. A plain C code of this model was downloaded from *cellML* [14] and used in our numerical experiments.

20

Chapter 3

NUMERICAL ALGORITHMS

Assuming homogeneous Neumann boundary condition along the boundary, i.e. zero normal derivative of voltage, the mathematical model describing electrical propagation in tissue occupying a region Ω is an initial-boundary value problem (IBVP) for a system of differential equations,

$$
\begin{cases}
C_m \frac{\partial}{\partial t} V(x,t) = \nabla \cdot \left(\frac{1}{R_a} \nabla V(x,t) \right) - I_{ion}(V), \\[2ex]
\frac{dg_p}{dt} = \alpha_{g_p}(V)(1 - g_p) - \beta_{g_p}(V)g_p \\[2ex]
\frac{\partial V}{\partial n} = 0, \forall x \in \partial \Omega \qquad \text{(Boundary Condition)} \\[2ex]
V(x,0) = V_0(x), \ g_m(0) = g_{p0}, \qquad \text{(Initial Condition)} \\[2ex]
x \in \Omega, \text{ a bounded domain in } R^d, d = 1,2,3, \quad 0 \le t \le T.
\end{cases} \tag{3.1}
$$

where g_p's are gate variables in the ionic model ($p = 1, 2, 3$ for Hudgkin-Huxley model and $p = 1, ..., 7$ for Luo-Rudy model).

The system of equations (3.1) consists of one PDE for voltage V and several ODEs for gate variables. We therefore discretize the PDE in space (by finite volume discretization) and rewrite (3.1) as a system of ODEs in time at spatial nodes.

By defining the state variable $\rho(x,t) = C_m V(x,t)$ and the flux $F(x,t) = -\frac{1}{R_a}\nabla V$, the PDE can be rewritten as

$$\frac{\partial}{\partial t}\rho(x,t) = -\nabla \cdot F(x,t) - I_{ion}(V).$$

Define the volume average of $\rho(x,t)$ in i^{th} control volume Ω_i at time t as

$$\overline{\rho}_i(t) = \frac{1}{|\Omega_i|}\int_{\Omega_i}\rho(x,t)d\Omega_i,$$

where $|\Omega_i|$ is the volume of Ω_i. Integrating the PDE over each control volume Ω_i, we have

$$\frac{d}{dt}\overline{\rho}_i(t) = -\frac{1}{|\Omega_i|}\int_{\Omega_i}\nabla \cdot F(x,t)d\Omega_i - \frac{1}{|\Omega_i|}\int_{\Omega_i}I_{ion}(V)d\Omega_i\,.$$

Now, we apply the divergence theorem, $\int_{\Omega}\nabla \cdot F d\Omega = \int_S F \cdot n dS$, and substitute the volume integral of the divergence with the normal component of $F(x,t)$ evaluated at the surface of the finite volume Ω_i to obtain a semi-discrete numerical scheme:

$$\frac{d}{dt}\overline{\rho}_i(t) = -\frac{1}{|\Omega_i|}\int_{\partial\Omega_i}F(x,t) \cdot \nabla n dS - \frac{1}{|\Omega_i|}\int_{\Omega_i}I_{ion}(V)d\Omega_i\,.$$

or equivalently,

$$C_m\frac{d}{dt}\overline{V}_i(t) = -\frac{1}{|\Omega_i|}\int_{\partial\Omega_i}F(x,t) \cdot \overrightarrow{n}dS - \frac{1}{|\Omega_i|}\int_{\Omega_i}I_{ion}(V)d\Omega_i\,. \qquad (3.2)$$

Notice that equation (3.2) is exact for the volume averages, i.e. no approximations have been made during its derivation.

Now we approximate the voltage and I_{ion} at the center of control volume Ω_i (at time t) by $V_i(t) \approx \overline{V}_i$ and $I_{ion}(V_i) \approx \frac{1}{|\Omega_i|} \int_{\Omega_i} I_{ion}(V) d\Omega_i$, respectively. We have

$$C_m \frac{d}{dt} V_i(t) = -\frac{1}{|\Omega_i|} \int_{\partial\Omega_i} F(x,t) \cdot \vec{n} dS - I_{ion}(V_i). \tag{3.3}$$

To compute the flux term on the right hand side of equation (3.3), for simplicity, we choose uniform rectanglular control volumes $\Omega_i = \Delta x \times \Delta y \times \Delta z$. Then we have

$$\text{1D:} \quad \frac{dV_i}{dt} = \frac{1}{C_m} \left[\frac{F_{i-\frac{1}{2}} - F_{i+\frac{1}{2}}}{\Delta x} - I(V_i) \right],$$

$$\text{2D:} \quad \frac{dV_{i,j}}{dt} = \frac{1}{C_m} \left[\frac{F_{i-\frac{1}{2},j} - F_{i+\frac{1}{2},j}}{\Delta x} + \frac{F_{i,j-\frac{1}{2}} - F_{i,j+\frac{1}{2}}}{\Delta y} - I(V_{i,j}) \right],$$

$$\tag{3.4}$$

$$\text{3D:} \quad \frac{dV_{i,j,k}}{dt} = \frac{1}{C_m} \left[\frac{F_{i-\frac{1}{2},j,k} - F_{i+\frac{1}{2},j,k}}{\Delta x} + \frac{F_{i,j-\frac{1}{2},k} - F_{i,j+\frac{1}{2},k}}{\Delta y} + \frac{F_{i,j,k-\frac{1}{2}} - F_{i,j,k+\frac{1}{2}}}{\Delta z} - I(V_{i,j,k}) \right],$$

where i, j, k are indices in $x, y, z-$directions, and $F_{(i,j,k)\pm\frac{1}{2}}$ are corresponding fluxes at the left and right faces, respectively. In simplified indexing, the (diffusive) fluxes are:

$$F_{i\pm\frac{1}{2}} = \frac{1}{R_{a_i\pm\frac{1}{2}}} \frac{V_{i-1} - V_i}{\Delta x}, \ F_{j\pm\frac{1}{2}} = \frac{1}{R_{a_j\pm\frac{1}{2}}} \frac{V_{j-1} - V_j}{\Delta y}, \ F_{k\pm\frac{1}{2}} = \frac{1}{R_{a_k\pm\frac{1}{2}}} \frac{V_{k-1} - V_k}{\Delta z}.$$

Similarly, applying the ODEs for the gate variables in (3.1) at each control volume yields

$$\frac{dg_p}{dt} = \alpha_{g_p}(V_{i,j,k})(1 - g_p) - \beta_{g_p}(V_{i,j,k})g_p. \tag{3.5}$$

23

Thus, a semi-discrete problem for (3.1) on each control volume $\Omega_{i,j,k}$ is given by (with simplified indexing for fluxes)

$$
\begin{cases}
\frac{dV_{i,j,k}}{dt} = \frac{1}{C_m}\left[\frac{F_{i-\frac{1}{2}}-F_{i+\frac{1}{2}}}{\Delta x} + \frac{F_{j-\frac{1}{2}}-F_{j+\frac{1}{2}}}{\Delta y} + \frac{F_{k-\frac{1}{2}}-F_{k+\frac{1}{2}}}{\Delta z} - I(V_{i,j,k})\right], \\[4mm]
\frac{dg_p}{dt} = \alpha_{g_p}(V_{i,j,k})(1-g_p) - \beta_{g_p}(V_{i,j,k})g_p \\[4mm]
V_x(0,y,z,t) = V_y(x,0,z,t) = V_z(x,y,0,t) = 0, \quad \text{(Boundary Conditions)} \\[4mm]
V(x,y,z,0) = V_0, \; g_p(0) = 0, \quad \text{(Initial Conditions)} \\[4mm]
(x,y,z) \in [0,L_x] \times [0,L_y] \times [0,L_z], \; 0 \le t \le T.
\end{cases}
\tag{3.6}
$$

We employ the following time stepping schemes to solve equation (3.6):

- Super-Time-Stepping Scheme, described in §3.1,

- DuFort-Frankel Scheme, described in §3.2,

- Runge-Kutta Schemes, explicit and implicit, of various orders, described in §3.3.

3.1   Super-Time-Stepping (STS) Scheme

Super time-stepping is a simple method to accelerate explicit schemes for parabolic problems [3]. In this section, we first state the idea of super time-stepping and then establish the stability and convergence of the scheme.

To simplify notation, let us rewrite the first semi-discrete equation in (3.6) in vector form as

$$
\frac{d\overrightarrow{V}}{dt}(t) + A\overrightarrow{V}(t) = \overrightarrow{f}(\overrightarrow{V}) \qquad t > 0 \qquad \overrightarrow{V}(0) = \overrightarrow{V}_0.
\tag{3.7}
$$

24

Applying the standard Forward Euler explicit scheme on (3.7) yields

$$\vec{V}^{n+1} = \vec{V}^n - \Delta t A \vec{V}^n + \Delta t \vec{f}(\vec{V}^n) = (I - \Delta t A)\vec{V}^n + \Delta t \vec{f}(\vec{V}^n)$$

$$n = 0, 1, \dots \qquad \vec{V}^0 = \vec{V}_0,$$

where $\vec{V}$ is a vector of values of voltage on all control volumes, Δt is the time step, $\vec{V}_0$ is the vector of the given initial values, and A is a symmetric positive definite matrix representing the discretized Laplace operator together with the boundary conditions. For example, in the one dimentsional case,

$$A = \left(\frac{1}{C_m \Delta x^2}\right)\begin{bmatrix} 1 & -1 & 0 & \cdots & 0 & 0 \\ -1 & 2 & -1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \cdots & \cdots \\ 0 & 0 & 0 & \cdots & 2 & -1 \\ 0 & 0 & 0 & \cdots & -1 & 1 \end{bmatrix}$$

The time step in Forward Euler scheme is subject to the restrictive stability condition (the famous Courant-Friedrichs-Lewy (CFL) stability condition)

$$\rho(I - \Delta t A) < 1 \implies \Delta t < \Delta t_{expl} = \frac{2}{\lambda_{max}},$$

where $\rho(\cdot)$ denotes the spectral radius and λ_{max} stands for the largest eigenvalue of A.

STS relaxes restriction of the CFL condition by requiring stability at the end of a cycle of N time steps, rather than at the end of each time step Δt, thus leading to a Runge-Kutta-like method with N stages. A *superstep* ΔT, consisting of N *substeps* $\tau_1, \dots, \tau_N$, $\Delta T = \sum_{j=1}^{N} \tau_j$, is introduced in the scheme. The idea is to achieve stability at the end of the superstep ΔT as well as maxmize the duration of the superstep.

25

The new scheme can be written as

$$\overrightarrow{V}^{n+1} = \left(\prod_{j=1}^{N} (I - \tau_j A) \right) \overrightarrow{V}^n + \sum_{j=1}^{N} (I - \tau_{j+1} A)...(I - \tau_N A) \tau_j \overrightarrow{f}(\overrightarrow{V}^{n,j}),$$

for $n = 1, 2, ...$ \hfill (3.8)

where $V^{n,j}$ denotes the computed voltage at time $n \Delta T + \sum_{k=1}^{j} \tau_k$. Note that $||I - \tau_j A|| \leq 1$ for all $j = 1, ..., N$. Assuming f is bounded in $\Omega \times [0, T]$ (which is true by the construction of ionic models), the second term on the right hand of (3.8) is bounded

$$|| \sum_{j=1}^{N} (I - \tau_j A)...(I - \tau_N A) \tau_j \overrightarrow{f}(\overrightarrow{V}^{n,j})|| \leq ||f||_\infty \sum_{j=1}^{N} \tau_j = |f|_\infty \Delta T .$$

Therefore, the corresponding stability condition for (3.8) is

$$\rho \left(\prod_{j=1}^{N} (I - \tau_j A) \right) < 1,$$

which is the same as that described in [3]. This relation is satisfied if

$$\left| \prod_{j=1}^{N} (I - \tau_j \lambda) \right| < 1 \quad \forall \lambda \in [\lambda_{min}, \lambda_{max}].$$

To obtain 'strong' stability, we replace above condition by

$$\left| \prod_{j=1}^{N} (I - \tau_j \lambda) \right| \leq K \quad \forall \lambda \in [\mu, \lambda_{max}],$$

where μ is some number in the interval $(0, \lambda_{max}]$, and K is some number between 0 and 1. The problem of finding the 'optimal' values for the τ_j's can be then reformulated

26

as [3]

$$\text{Find } \tau_1, \tau_2, ..., \tau_N \text{ such that } p_N(\lambda) = \prod_{j=1}^{N}(1 - \tau_j\lambda) \text{ satisfies}$$

$$|p_N(\lambda)| \leq K \qquad \forall \lambda \in [\mu, \lambda_{max}] \qquad \text{(stability), and}$$

$$|p'_N(0)| = \sum_{j=1}^{N} \tau_j \qquad \text{maximal} \qquad \text{(optimality)}.$$

It is also pointed out in [3] that using the optimality properties of the Chebyshev polynomials $T_N(\cdot)$ of degree N, the optimal values of the τ_j's are those for which

$$p_N(\lambda) = T_N\left(\frac{\lambda_{max} + \mu - 2\lambda}{\lambda_{max} - \mu}\right) \Big/ T_N\left(\frac{\lambda_{max} + \mu}{\lambda_{max} - \mu}\right),$$

provided

$$K = 1/T_N\left(\frac{\lambda_{max} + \mu}{\lambda_{max} - \mu}\right) \leq 1.$$

Note that K may be chosen arbitrarily close to 1 by choosing μ small enough. The substeps τ_j's corresponding to the above polynomial p_N are given explicitly by

$$
\begin{aligned}
\tau_j &= 2\left((-\lambda_{max} + \mu)\cos\left(\frac{2j-1}{N}\frac{\pi}{2}\right) + \lambda_{max} + \mu\right)^{-1} \\
&= \Delta t_{expl}\left((-1 + \nu)\cos\left(\frac{2j-1}{N}\frac{\pi}{2}\right) + 1 + \nu\right)^{-1}, \quad j = 1, ..., N, \quad (3.9)
\end{aligned}
$$

where $0 < \nu = \mu/\lambda_{max} < \lambda_{min}/\lambda_{max}$.

Theorem 3.1. *(Stability)*

If $||f||_\infty < \infty$, the super-time-stepping scheme for problem (3.6) is stable at the end of every superstep consisting of N substeps given in (3.9).

Moreover, one can also show that

$$\Delta T = \sum_{j=1}^{N} \Delta \tau_j = \Delta t_{\text{expl}} \frac{N}{2\sqrt{\nu}} \left[\frac{(1 + \sqrt{\nu})^{2N} - (1 - \sqrt{\nu})^{2N}}{(1 + \sqrt{\nu})^{2N} + (1 - \sqrt{\nu})^{2N}} \right], \qquad (3.10)$$

which yields

$$\Delta T \to N^2 \Delta t_{\text{expl}} \quad as \quad \nu \to 0. \qquad (3.11)$$

Equation (3.11) shows that super-time-stepping is (up to) N times faster than the standard explicit scheme. The speed-up comes from that a superstep consisting of N substeps covers a time interval, at essentially the same cost, N times longer (when $\nu \approx 0$) than time $N\Delta t_{expl}$, which is covered by N explicit steps, each of length Δt_{expl}.

Only the values at the end of each superstep approximate the solution of the problem since STS ensures stability only at the end of each superstep. STS reduces to plain Forward Euler by setting the parameters as $N = 1$, $\nu = 0$.

In addition to speeding up the computation, the super-time-stepping scheme is extremely simple to implement in any existing explicit code. As we will see in §3.5, it turns out to be, by far, the fastest of all eleven solvers we tested.

Theorem 3.2. *(Convergence)*

The numerical solution obtained by the super-time-stepping scheme on problem (3.6) converges to the exact solution provided $||f||_\infty < \infty$, f is Lipschitz continuous with respect to V, and V Lipschitz continuous in t.

Proof. Using the semigroup approach and noting that $\Delta T = \sum_{j=1}^{N} \tau_j$, the exact solution of (3.6) at time $t = k\Delta T$ can be written in integral form as

$$
\begin{aligned}
\overrightarrow{V}(k\Delta T) &= e^{-\Delta T A} \overrightarrow{V}((k-1)\Delta T) + \int_{(k-1)\Delta T}^{k\Delta T} e^{-(k\Delta T - \tau)A} f(\overrightarrow{V}(\tau)) d\tau \\
&= e^{-\Delta T A} \overrightarrow{V}((k-1)\Delta T) + \sum_{j=1}^{N} \int_{(k-1)\Delta T + \sum_{i=1}^{j-1} \tau_i}^{(k-1)\Delta T + \sum_{i=1}^{j} \tau_i} e^{-(k\Delta T - \tau)A} f(\overrightarrow{V}(\tau)) d\tau.
\end{aligned}
$$

28

Therefore, the error between the exact and approximate solutions is given by

$$
\begin{aligned}
||E^k|| &= ||\overrightarrow{V}(k\Delta T) - \overrightarrow{V}^k|| \\
&= \left\| \left(e^{-\Delta T A}\overrightarrow{V}((k-1)\Delta T) + \int_{(k-1)\Delta T}^{k\Delta T} e^{-(k\Delta T-\tau)A} f(\overrightarrow{V}(\tau))d\tau \right) - \right. \\
&\quad \left. \left(\left(\prod_{j=1}^{N}(I-\tau_j A) \right) \overrightarrow{V}^n + \sum_{j=1}^{N}(I-\tau_{j+1}A)...(I-\tau_N A)\tau_j f(\overrightarrow{V}^{n,j}) \right) \right\| \\
&\leq \left\| e^{-\Delta T A}\overrightarrow{V}((k-1)\Delta T) - \prod_{j=1}^{N}(I-\tau_j A)\overrightarrow{V}^n \right\| + \\
&\quad \left\| \int_{(k-1)\Delta T}^{k\Delta T} e^{-(k\Delta T-\tau)A} f(\overrightarrow{V}(\tau))d\tau - \sum_{j=1}^{N}(I-\tau_{j+1}A)...(I-\tau_N A)\tau_j f(\overrightarrow{V}^{n,j}) \right\| \\
&\leq \left\| e^{-\Delta T A} - \prod_{j=1}^{N}(I-\tau_j A)V^{k-1} \right\| ||V((k-1)\Delta T)|| + \\
&\quad \left\| \prod_{j=1}^{N}(I-\tau_j A) \right\| ||V((k-1)\Delta T) - V^{k-1}|| + \\
&\quad \sum_{j=1}^{N} \int_{(k-1)\Delta T+\sum_{i=1}^{j-1}\tau_i}^{(k-1)\Delta T+\sum_{i=1}^{j}\tau_i} \left\| e^{-(k\Delta T-\tau)A} f(\overrightarrow{V}(\tau)) - (I-\tau_{j+1}A)...(I-\tau_N A)f(\overrightarrow{V}^{n,j}) \right\| d\tau
\end{aligned}
$$

Since A is positive definite, we have

$$
\begin{aligned}
||V((k-1)\Delta T)|| &\leq ||e^{-(k-1)\Delta T A}V_0|| + ||f||_\infty ||A^{-1}e^{-(k-1)\Delta T A}|| \\
&\leq ||V_0|| + C.
\end{aligned}
$$

On the other hand, we notice that A being symmetric means that $\prod_{j=1}^{N}(I-\tau_j A)$ is also symmetric, and thus the stability condition is equivalent to $||\prod_{j=1}^{N}(I-\tau_j A)|| < 1$.

Then, by Lipschitz continuity of f and V, we have

$$\sum_{j=1}^{N} \int_{(k-1)\Delta T + \sum_{i=1}^{j-1} \tau_i}^{(k-1)\Delta T + \sum_{i=1}^{j} \tau_i} \left\| e^{-(k\Delta T - \tau)A} f(\overrightarrow{V}(\tau)) - (I - \tau_{j+1} A)...(I - \tau_N A) f(\overrightarrow{V}^{n,j}) \right\| d\tau$$

$$\leq C \sum_{j=1}^{N} \int_{(k-1)\Delta T + \sum_{i=1}^{j-1} \tau_i}^{(k-1)\Delta T + \sum_{i=1}^{j} \tau_i} \left\| f(\overrightarrow{V}(\tau)) - f(\overrightarrow{V}^{n,j}) \right\| d\tau$$

$$\leq C \sum_{j=1}^{N} \int_{(k-1)\Delta T + \sum_{i=1}^{j-1} \tau_i}^{(k-1)\Delta T + \sum_{i=1}^{j} \tau_i} (\tau - (k-1)\Delta T) d\tau$$

$$\leq C \sum_{j=1}^{N} (\tau_1 + \tau_2 + ... + \tau_j)^2$$

$$\leq C N \Delta T^2.$$

Putting all these together yields

$$\|E^k\| \leq \left\| e^{-\Delta T A} - \prod_{j=1}^{N} (I - \tau_j A) V^{k-1} \right\| (\|V_0\| + C) + \|E^{k-1}\| + C N \Delta T^2$$

$$\leq k \left\| e^{-\Delta T A} - \prod_{j=1}^{N} (I - \tau_j A) V^{k-1} \right\| (\|V_0\| + C) + k C N \Delta T^2.$$

Expanding and keeping only the lowest-order terms, we obtain

$$\|E^k\| \leq \left(k \frac{\lambda_{max}}{2} \|V_0\| + C \right) \Delta T^2.$$

As expected, the method is essentially of order one w.r.t ΔT. $\qquad \square$

It is worth noting that, although we justify the method only in case the operator A appearing in (3.6) is a symmetric positive definite matrix, such an assumption does not appear to be always required in practice.

3.2 DuFort-Frankel (DF) Scheme

DuFort-Frankel, a modication of the Leapfog scheme, is an explicit, 2-step, second order accurate in space and time, theoretically unconditionally stable scheme [36].

It is obtained as follows. Applying centered finite difference in space and Forward Euler in time on the first equation in (3.4) results in

$$C_m \frac{V_i^{n+1} - V_i^n}{\Delta t} = \frac{1}{\Delta x^2} \left(\frac{V_{i-1}^n - V_i^n}{R_{a\,i-\frac{1}{2}}} - \frac{V_i^n - V_{i+1}^n}{R_{a\,i+\frac{1}{2}}} \right) + I_{ion}(V_i^n).$$

This difference scheme is explicit but conditionally stable. It can be stabilized by replacing the V_i^n term with the average over two time steps $(V_i^{n+1} + V_i^{n-1})/2$, producing a 2-step scheme.

$$C_m \frac{V_i^{n+1} - V_i^{n-1}}{2\Delta t} = \frac{1}{\Delta x^2} \left[\frac{V_{i-1}^n - \frac{1}{2}\left(V_i^{n+1} + V_i^{n-1}\right)}{R_{a\,i-\frac{1}{2}}} + \frac{V_{i+1}^n - \frac{1}{2}\left(V_i^{n+1} + V_i^{n-1}\right)}{R_{a\,i+\frac{1}{2}}} \right]$$
$$- I_{ion}\left(V_i^n\right).$$

The resulting difference equation is a three time level expression which has the advantages of unconditional stability and second order accuracy in both time and space [36].

However, numerical experiments show that small oscillations occur near the steady state. To avoid this and keep the scheme explicit, the average of voltage at previous two time steps is used to evaluate the ionic current I_{ion} and yields,

$$C_m \frac{V_i^{n+1} - V_i^{n-1}}{2\Delta t} = \frac{1}{\Delta x^2} \left[\frac{V_{i-1}^n - \frac{1}{2}\left(V_i^{n+1} + V_i^{n-1}\right)}{R_{a\,i-\frac{1}{2}}} + \frac{V_{i+1}^n - \frac{1}{2}\left(V_i^{n+1} + V_i^{n-1}\right)}{R_{a\,i+\frac{1}{2}}} \right]$$
$$- I_{ion}\left(\overline{V}_i^n\right), \tag{3.12}$$

where $\overline{V}_i^n = (V_i^n + V_i^{n-1})/2$. This is the DF numerical scheme for the 1D cable equation.

Using the fact that the space directions are orthogonal, we can easily extend scheme (3.12) to the three dimensional DuFort-Frankel difference scheme for the semi-discrete cable equation in (3.6)

$$
\begin{aligned}
&C_m \frac{V_{i,j,k}^{n+1} - V_{i,j,k}^{n-1}}{2\Delta t} \\
&= \frac{1}{\Delta x^2} \left[\frac{V_{i-1,j,k}^n - \frac{1}{2}\left(V_{i,j,k}^{n+1} + V_{i,j,k}^{n-1}\right)}{R_{a\,i-\frac{1}{2}},j,k} + \frac{V_{i+1,j,k}^n - \frac{1}{2}\left(V_{i,j,k}^{n+1} + V_{i,j,k}^{n-1}\right)}{R_{a\,i+\frac{1}{2},j,k}} \right] \\
&+ \frac{1}{\Delta y^2} \left[\frac{V_{i,j-1,k}^n - \frac{1}{2}\left(V_{i,j,k}^{n+1} + V_{i,j,k}^{n-1}\right)}{R_{a\,i,j-\frac{1}{2}},k} + \frac{V_{i,j+1,k}^n - \frac{1}{2}\left(V_{i,j,k}^{n+1} + V_{i,j,k}^{n-1}\right)}{R_{a\,i,j+\frac{1}{2},k}} \right] \\
&+ \frac{1}{\Delta z^2} \left[\frac{V_{i,j,k-1}^n - \frac{1}{2}\left(V_{i,j,k}^{n+1} + V_{i,j,k}^{n-1}\right)}{R_{a\,i,,j,k-\frac{1}{2}}} + \frac{V_{i,j,k+1}^n - \frac{1}{2}\left(V_{i,j,k}^{n+1} + V_{i,j,k}^{n-1}\right)}{R_{a\,i,j,k+\frac{1}{2}}} \right] \\
&- I_{ion}\left(\overline{V}_i^n\right).
\end{aligned}
\tag{3.13}
$$

On the other hand, the ODEs for the gates are discretized by Forward Euler, and again evaluated at the average of the two previous voltage values,

$$
\frac{(g_p)_{i,j,k}^{n+1} - (g_p)_{i,j,k}^n}{\Delta t} = \alpha_{g_p}\left(\overline{V}_i^n\right)\left(1 - (g_p)_{i,j,k}^n\right) - \beta_{g_p}\left(\overline{V}_i^n\right)(g_p)_{i,j,k}^n.
\tag{3.14}
$$

Equations (3.13) and (3.14) together give the scheme.

Being aware of when the stimulus takes place, a time-step factor $dtfac$ is introduced to speed up the computation here. That is, Δt_{expl} (satisfying the CFL condition) is used in a small time interval containing the moment stimulus happens, and a larger time step $\Delta t_{big} = dtfac * \Delta t_{expl}$ is used elsewhere. In the simulations, we used $dtfac = 1$ and $dtfac = 2$, denoting the schemes as DF1 and DF2. They produce similar results.

Stability, Consistency and Convergence Results

To prove convergence of the nonhomogeneous differnce scheme, it is enough to establish stability of the homogeneous equation, along with the correct consistency. All of the contributions of the nonhomogeneous term will be contained in the

truncation error term. Thus, when we discuss stability of a nonhomogeneous difference scheme (3.12) or (3.13), we consider the stability of the associated homogeneous scheme.

The homogeneous equation of the one dimensional cable equation is a diffusion equation, which for simplicity we write as $V_t = bV_{xx}$. The associated DuFort-Frankel scheme is

$$V_i^{n+1} - V_i^{n-1} = 2b\mu(V_{i+1}^n - (V_i^{n+1} + V_i^{n-1}) + V_{i-1}^n),$$

where $\mu = \Delta t/\Delta x^2$. Rearranging the terms yields

$$(1 + 2b\mu)V_i^{n+1} - (1 - 2b\mu)V_i^{n-1} = 2b\mu(V_{i+1}^n + V_{i-1}^n).$$

To study stability via the von Neumann approach, we substitute $V_i^n = g^n e^{\jmath m\Delta x\xi}$, where $\jmath = \sqrt{-1}$ is the *imaginary unit*, to get

$$(1 + 2b\mu)g^2 - (1 - 2b\mu) = 2b\mu\left(e^{\jmath\Delta x\xi} + e^{-\jmath\Delta x\xi}\right)g,$$

which implies

$$g_\pm = \frac{2b\mu\cos(\Delta x\xi) \pm \sqrt{1 - 4b^2\mu^2\sin^2(\Delta x\xi)}}{1 + 2b\mu}.$$

The scheme is not dissipative since $g_-(\pi) = -1$. For stability, we need to show that $|g| \leq 1$ for all ξ.

Now, if $1 - 4b^2\mu^2\sin^2(\Delta x\xi) \geq 0$, then $|g_\pm| \leq \frac{2b\mu|\cos(\Delta x\xi)|+1}{1+2b\mu} \leq \frac{2b\mu+1}{1+2b\mu} = 1$. On the other hand, if $1 - 4b^2\mu^2\sin^2(\Delta x\xi) < 0$, then $|g_\pm|^2 = \frac{(2b\mu\cos(h\xi))^2 + 4b^2\mu^2\cos^2(h\xi)-1}{1+2b\mu} = \frac{4b^2\mu^2-1}{(1+2b\mu)^2} = \frac{2b\mu-1}{1+2b\mu} \leq 1$. Finally, when $1 - 4b^2\mu^2\sin^2(\Delta x\xi) = 0$, then $|g_\pm| \leq \frac{2b\mu|\cos(\Delta x\xi)|}{1+2b\mu} < 1$.

Thus, we have stability for any value of μ, so the DuFort-Frankel scheme is unconditionally stable.

Next we examine consistency. We rewrite the scheme as

$$\frac{V_i^{n+1} - V_i^{n-1}}{2\Delta t} = b\frac{V_{i+1}^n - 2V_i^n + V_{i-1}^n}{\Delta x^2} - b\frac{V_i^{n+1} + V_i^{n-1} - 2V_i^n}{\Delta x^2},$$

and by Taylor expansions we see that it approximates

$$V_t + \frac{\Delta t^2}{6}V_{ttt} = b\left(V_{xx} + \frac{\Delta x^2}{12}V_{xxxx}\right) - b\left(\frac{\Delta t^2}{\Delta x^2}V_{tt} + \frac{\Delta t^4}{12\Delta x^2}V_{tttt}\right).$$

Assuming V_{tt}, V_{ttt}, V_{tttt}, and V_{xxxx} remain bounded as Δx, $\Delta t \to 0$, the scheme approximates the Telegrapher's equation $V_t = bV_{xx} - b(\Delta t/\Delta x)^2V_{tt}$. It will be consistent with the diffusion equation $V_t = bV_{xx}$ only if $\Delta t/\Delta x \to 0$ as $\Delta x, \Delta t \to 0$.

At the same time, we see that the order of the scheme is $O(\Delta t^2 + \Delta x^2 + \Delta t^2/\Delta x^2)$, which is dominanted by $\Delta t^2/\Delta x^2$, provided V_{tt}, V_{ttt}, V_{tttt}, and V_{xxxx} remain bounded.

Finally, we consider the scheme (3.14) applied on the ODEs for the gates and show that Δt must be restricted by certain Δt_{cutoff} value to maintain stability. Rewrite (3.14) as

$$\begin{aligned}(g_p)_{i,j,k}^{n+1} &= (g_p)_{i,j,k}^n + \Delta t\left(\alpha_{g_p}\left(\overline{V}_i^n\right)\left(1 - (g_p)_{i,j,k}^n\right) - \beta_{g_p}\left(\overline{V}_i^n\right)(g_p)_{i,j,k}^n\right)\\ &= \left(1 - \Delta t\left(\alpha_{g_p}\left(\overline{V}_i^n\right) + \beta_{g_p}\left(\overline{V}_i^n\right)\right)\right)(g_p)_{i,j,k}^n + \Delta t\alpha_{g_p}\left(\overline{V}_i^n\right).\end{aligned}$$

Stability is guaranteed if

$$|1 - \Delta t(\alpha_{g_p} + \beta_{g_p})| \leq 1 \tag{3.15}$$

for all gates g_p. In ionic models, α_{g_p}'s and β_{g_p}'s, expressed explicitly by messy equations of voltage V, are bounded since they all depend continuously on V in a physical range of $[\widehat{V}_{min}, \widehat{V}_{max}]$. Thus, the inequality (3.15) holds for small $\Delta t < 1$.

In our simulations, a strong external stimulus I_{stim} is applied to certain small region of the cable, during a short time, to stimulate propagation of the voltage V. This makes V in the stimulated region fall into a bigger range $[V_{min}, V_{max}]$ than the ideal range $[\widehat{V}_{min}, \widehat{V}_{max}]$. Therefore, we have to restrict the time step Δt to be smaller.

We therefore define

$$\Delta t_{cutoff} := \min \left\{ \frac{2}{\alpha_{g_p} + \beta_{g_p}} \right\} < 1, \quad \forall V \in [V_{min}, V_{max}] \text{ and all } g_p\text{'s}, \qquad (3.16)$$

and require $\Delta t \leq \Delta t_{cutoff}$ for stability. It turns out that the value $\Delta t_{cutoff} = 0.01\,ms$ is sufficient for stability of the ODEs of the Luo-Rudy model. This restriction is applied to time steps of the Dufort-Frankel scheme, as well as to time steps of all the other time-steppers we use.

Summarizing our discussion above yields the next theorem

Theorem 3.3. *The scheme (3.13-3.14) is stable if the time step $\Delta t \leq \Delta t_{cutoff}$. It is consistent with the diffusion equation when $\Delta t/\Delta x \to 0$ as Δt, $\Delta x \to 0$. Moreover, the scheme is of order $O(\Delta t^2 + \Delta x^2 + \Delta t^2/\Delta x^2)$ provided V_{tt}, V_{ttt}, V_{tttt}, and V_{xxxx} remain bounded.*

3.3 Runge-Kutta (RK) Schemes

Runge-Kutta is a large family of methods for the numerical solution of ODE systems

$$\frac{dy}{dt} = f(t, y), \quad y(t_0) = y_0.$$

These methods can be expressed in the form [60]

$$y_{n+1} = y_n + \sum_{i=1}^{s} b_i k_i$$

$$k_i = f\left(t_n + c_i h, y_n + h \sum_{j=1}^{i-1} a_{ij} k_j\right)$$

They are single-step, multi-stage methods, with s stages $k_i, i = 1, ..., s$. Each method can be described by a *Butcher tableau*, which puts the coefficients of the method in a table as follows:

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

The family of Runge-Kutta methods includes low and high order, explict and implicit, non-adaptive and adaptive integrators.

We tested the following non-adaptive, explicit and implicit methods:

- Classical explicit fourth-order method (RK4)

- Implicit second-order method (RK2imp)

- Implicit fourth-order method (RK4imp)

In additon, we tested several adaptive embedded Runge-Kutta methods. Embedded methods are designed to produce an estimate of the local truncation error of a single Runge-Kutta step, and as result, allow control of the error via adaptive stepsize. This is done by computing with two s-stage methods, of orders p and p-1, which use the same values for the stages k_i thus avoiding additional computational cost. Representing the lower-oder step as

$$
y_{n+1}^* = y_n + h \sum_{i=1}^{s} b_i^* k_i,
$$

with k_i same as for the higher order method, then the error can be estimated as

$$
e_{n+1} = y_{n+1} - y_{n+1}^* = h \sum_{i=1}^{s} (b_i - b_i^*) k_i,
$$

which is of order $O(h, p)$. The Butcher Tableau for this kind of method is

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

The adaptive embedded methods we use are the following:

- The Bogacki-Shampine method (RK23) [8]

$$
\begin{array}{c|cccc}
0 & & & & \\
1/2 & 1/2 & & & \\
3/4 & 0 & 3/4 & & \\
1 & 2/9 & 1/3 & 4/9 & \\
\hline
 & 2/9 & 1/3 & 4/9 & 0 \\
 & 7/24 & 1/4 & 1/3 & 1/8
\end{array}
$$

- The Runge-Kutta-Fehlberg(4,5) method (RK45) [16]

$$
\begin{array}{c|cccccc}
0 & & & & & & \\
1/4 & 1/4 & & & & & \\
3/8 & 3/32 & 9/32 & & & & \\
12/13 & 1932/2197 & -7200/2197 & 7296/2197 & & & \\
1 & 439/216 & -8 & 3680/513 & -845/4104 & & \\
1/2 & -8/27 & 2 & -3544/2565 & 1859/4104 & -11/40 & \\
\hline
 & 25/216 & 0 & 1408/2565 & 2197/4104 & -1/5 & 0 \\
 & 16/135 & 0 & 6656/12825 & 28561/56430 & -9/50 & 2/55
\end{array}
$$

- The Cash-Karp(4,5) method (RKCK) [13]

0					
1/5	1/5				
3/10	3/40	9/40			
3/5	3/10	-9/10	6/5		
1	-11/54	5/2	70/27	35/27	
7/8	1631/55296	175/512	575/13824	44275/110592	253/4096
	37/378	0	250/621	125/594	0	512/1771
	2825/27648	0	18575/48384	13525/55296	277/14336	1/4

- The Dormand-Prince(8,9) method (RK8PD) [15]

All these methods are efficiently implemented in the GNU Scientific Library (GSL 1.10) [21] from GNU, which we employ in our codes.

Stability of Explicit Runge-Kutta Methods

Consider the general case of p-stage Runge-Kutta method defined by the Butcher Tableau

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

Let Y be a vector made up from the p-stage values which satisfies

$$Y = y_0 + \Delta t\, AY = y_0 + zAY \,.$$

Solving for Y, gives $Y = (I - zA)^{-1} y_0$, from which we obtain

$$y_1 = y_0 + \Delta t\, \vec{b}^{\,T} Y = y_0 + z\, \vec{b}^{\,T} (I - zA)^{-1} y_0 = R(z)y_0,$$

where

$$R(z) = 1 + z\, \vec{b}^{\,T} (I - zA)^{-1}$$

is called the *stability function*. To achieve stability of the method, we want $|R(z)| < 1$. $R(z)$ can be evaluated as the exponential series truncated at the z^s term,

$$\overrightarrow{b}^T A^{k-1} 1 = \overrightarrow{b}^T A^{k-1} \overrightarrow{c} = \frac{1}{k!}, \quad k = 1, 2, ..., p.$$

Therefore, the stability function for each p is as follows [47]:

$$R(z) = \begin{cases} 1 + z, & p = 1 \\ 1 + z + \frac{1}{2}z^2, & p = 2 \\ 1 + z + \frac{1}{2}z^2 + \frac{1}{6}z^3, & p = 3 \\ 1 + z + \frac{1}{2}z^2 + \frac{1}{6}z^3 + \frac{1}{24}z^4, & p = 4 \\ \vdots \end{cases} \quad (3.17)$$

The stability regions of the stability functions in (3.17) are shown in Figure 3.1.
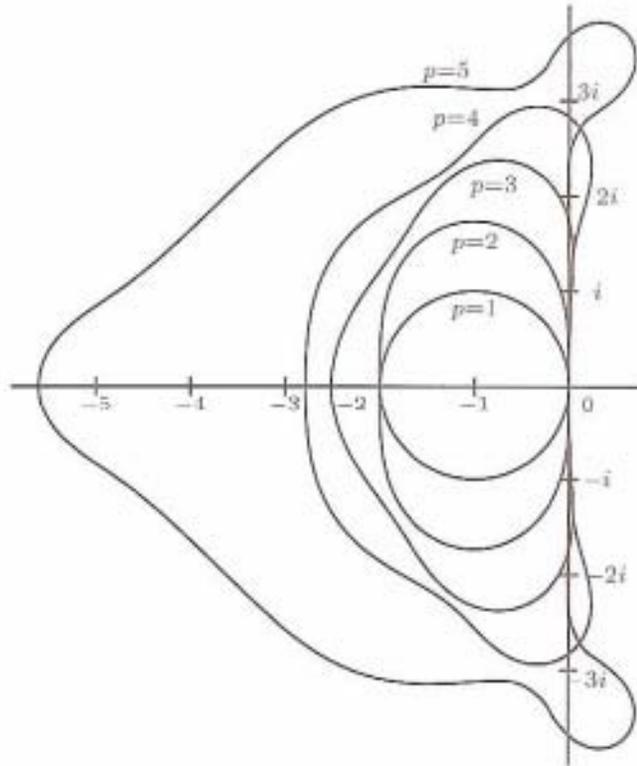
Figure 3.1: Stability regions of Explicit Runge-Kutta methods [47]

3.4   Library Method

It turns out that evaluating the ionic currents is very expensive and, as mentioned near the end of §3.2, it requires time-steps **no larger than** $0.01ms$. This penalizes all schemes, especially implicit and high order ones. In an attempt to reduce run times, we **pre-compute** all the $\alpha(V)$, $\beta(V)$ coefficients in the range $[-100, 200]$ with $\Delta V = 0.001$, and store them in a direct access, binary file (37 MB size), which is loaded into memory at run time. Then, values of $\alpha(V_k^n)$, $\beta(V_k^n)$ at any V_k^n are found by interpolation. This approach has come to be known as the **library method** [52].

In Figure 3.2 the timings of all eleven time integrators are compared without and with pre-computed library. Clearly, pre-computing has high payoff, reducing CPU time to almost half on each scheme, with no loss of accuracy (they produce the same propagation speed, V_{max}, $\{dV/dt\}_{max}$, and APD).

All other computations discussed here use the pre-computed library, whenever possible. A situation in which pre-computing is not feasible occurs in simulations with different value of the parameter $[K]_o$ in different regions, described in §5.1.5, §5.2.2 and §5.2.3. This parameter enters several of the coefficients $\alpha(V)$, $\beta(V)$, making pre-computing them impossible.

3.5   Comparison of Eleven Serial Solvers

Serial implementation results and comparison of eleven numerical schemes have been reported in our papers [28, 29]. These schemes were discussed in the previous sections of this chapter.

In these experiments a resistivity value of $Ri = 150 \; k\Omega cm$ was used, which is a thousand times larger than a realistic one, in order to speed up the computations, and they were still very long! With this higher resistivity, the action potential propagates much slower (with speed $\sim 3 \; cm/s$ instead of $\sim 100 \; cm/s$), so a single stimulus
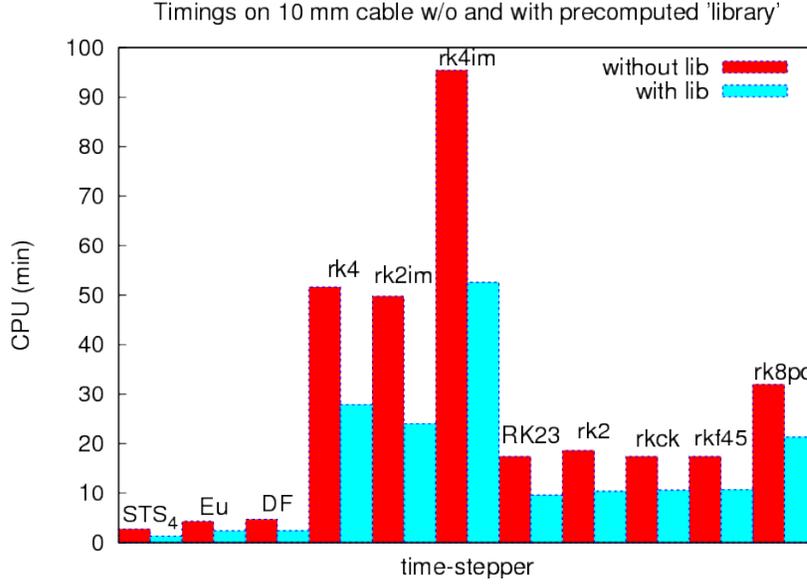
Figure 3.2: Timings of all eleven schemes on 10mm cable, without (red) and with precomputed library (cyan). Precomputing achieves almost 100% speedup on most schemes.

spreads much further apart along the cable, unlike the case with the physical value of $150\,\Omega cm$.

All these simulations were done on a $50\ mm$ cable, on the same machine, with normal parameter values (see Table 5.1), and mesh size $\Delta x = 8\ \mu m$, resuliing in 6250 control volumes. In all schemes, the time step Δt was restricted to be less than $0.01\ ms$ to ensure stability of the ODE system (see discussion near the end of §3.2).

Comparison of the timings of all eleven solvers is shown in Figure 3.3. Note that timings are in *minutes* of CPU time!

Our numerical experiments show that all the high order schemes produce identical voltage history (action potentials), and identical values for the biological quantities. Among them, the explicit adaptive 4th order solver **rkck** is the most efficient, followed closely by **rk45**. At the other end, the implicit non-adaptive 4th order **rk4imp** is by far the worst, with no redeeming features.

The low order schemes **STS, Eu, DF** are much faster than the high order ones, by factors of 10 to 25! , and **STS4** (i.e. STS with $N = 4$, $\nu = 0.07$) is the most efficient
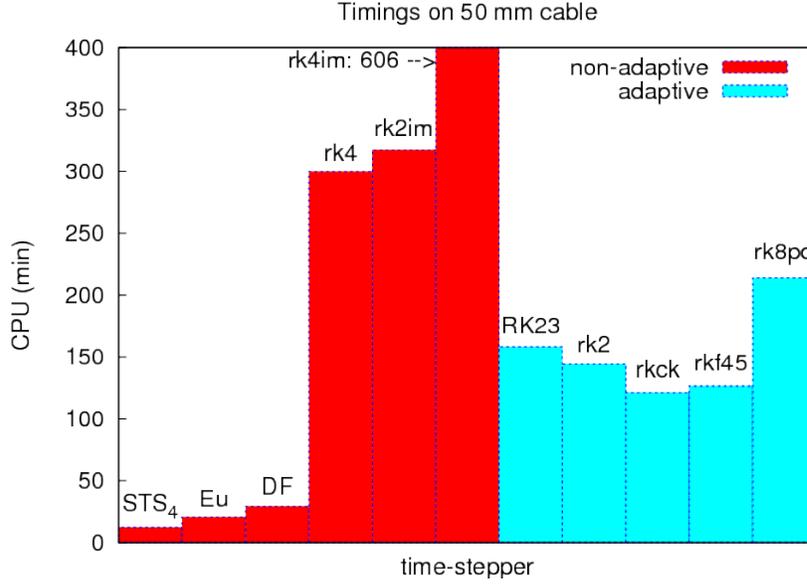
Figure 3.3: Timings of all eleven schemes on 50mm cable.
Non-adaptive (red) and adaptive (cyan) schemes.

of all. However, in the unphysical case $Ri = 150\,k\Omega\,cm$, they produce upstrokes somewhat delayed (by 20 - 50 ms at the end of $50mm$ cable), and slightly lower propagation speed than the high order ones. This does not happen in the physical case of $Ri = 150\,\Omega\,cm$.

Among high order solvers, the adaptive ones outperform the non-adaptive by a factor of 2 or more, and **rkck** is best among them. When adaptivity cannot be used, as is the case in parallel computations (Chapter 4), then **rk4** would be best among high order solvers.

In view of the fact that evaluation of the source restricts the time-step to $\Delta t \leq 0.01$, the implicit 2nd order solver **rk2imp** performs surprisingly well, being competitive with **rk4**, whereas rk4imp is hopelessly slow.

Chapter 4

PARALLEL COMPUTING

The difficulties in numerical simulations of propagation of action potentials arise primarily from the following:

- The size of, for instance, human heart cells, varies from 30 to 130 μm, whereas the length of a cardiac cell bundle ranges from 10 to 20 cm. Therefore the simulated domain consists of tens of thousands of biological cells, and each cell needs to be discretized by several numerical control volumes.

- The effective diffusion coefficient, $1/(C_m R_a)$, is high (of the order of $1\,cm^2/sec$). It requires very small time steps to resolve the fast evolution of the voltage.

These issues make the computational cost for a realistic simulation dramatically high. We therefore turn to *parallel computing* for help.

Parallel computing operates on the principle that large problems can often be divided into smaller ones, which are then solved concurrently ("in parallel") on many processors. There are several different levels of parallel computing: bit-level, instruction level, data, and task parallelism. Parallelism has been employed for many years, mainly in high-performance computing, but interest in it has grown since the late 1980's due to the physical constraints preventing frequency scaling on single processors. As power consumption (and consequently heat generation) by computers

has become a concern, parallel computing has become the dominant paradigm in computer architecture, mainly in the form of multicore processors and multiprocessor clusters [58].

4.1 Classical Performance Analysis

Parallel computing is a powerful tool to accelerate scientific computations. However, analyzing the performance of parallel programs is much more challenging than that of serial ones. We will introduce several commonly used quantities, as well as some important theories, for performance analysis of parallel algorithms before proceeding to analyze the performance of parallel algorithms applied in solving our problem. Definitions and theoretical results described in this section can be found in many resources. We refer interested readers to [2, 4, 23, 58] for more details.

4.1.1 Speedup

A core concept in parallel computing is **speedup**, which compares the execution of the parallel program with its serial cousin. Two types of speedup, absolute and relative, will be defined for completeness of this section but only the relative speedup will be used later.

Definition 4.1. *(Absolute Speedup [2, 23]) Let T_A be the wall clock time of the serial implementation and T_P be the wall clock time of the parallel implementation using P processors. The absolute speedup S_{P_A} is defined to be*

$$S_{P_A} = \frac{T_A}{T_P}.\tag{4.1}$$

The absolute speedup has significant theoretical meaning. However, it is difficult, even impossible, to measure in most cases. Absolute speed compares the parallel algorithm directly to the fastest serial one, which most likely is not available, and it

is highly affected by the implementation, machine architecture, compiler, etc. Thus, an alternative, the relative speed, is much more useful in practice.

Definition 4.2. *(Relative Speedup [23]) Let T_1 be the wall clock time of the code running on a single processor, and T_P again the wall clock time of the implementation over P processors. The relative speedup w.r.t one processor is defined as*

$$S_P^1 = \frac{T_1}{T_P}. \tag{4.2}$$

Similarly, if we let T_k be the wall clock time of the code running on $k < P$ processors and T_P the time on P processors, then a generalized relative speedup on k processors is defined as

$$S_P^k = \frac{T_k}{T_P}, \tag{4.3}$$

In this work we use relative speedup for all performance analysis results for convenience. To simplify notations, the relative speedup is denoted as S, S_P, or just speedup from now on.

The next defintion defines linear and super-linear speedups, which are the best we can expect for any parallel algorithm.

Definition 4.3. *Speedup is considered to be linear whenever*

$$S_P \approx P, \tag{4.4}$$

and it is called super linear whenever

$$S_P > P. \tag{4.5}$$

When neither one of these apply, the speedup is said to be nonlinear.

Linear speedup is the best we can expect for most parallel algorithms. Super-linear speedup is usually achieved from the improvement of hardware capability. For

instance, a common case of super-linear speedup, pointed out in [23], arises when large data sets cannot fit into single cache but can fit into multiple cashes when more processors are used. Consequently, provided the algorithm has linear speedup, combination of reduced memory access time and additional speedup may produce superlinear speedup.

Very few algorithms are capable of achieving linear, much less super-linear speedup, due to the fact that communication between processors, which can hardly be avoided, contributes more in the overhead and significantly slows down the computation.

Typically, good parallel algorithms achieve nearly linear speedup for small number of processors, which flattens out for large number of processors. As it will be demonstrated later, our parallel algorithms behave exactly this way.

4.1.2   Efficiency

The efficiency of an algorithm is another primary quantity for performance analysis.

Definition 4.4. *(Efficiency [2, 23])The efficiency of an algorithm using P processors is defined as*

$$E_P = \frac{S_P}{P}. \tag{4.6}$$

Thus, efficiency is speedup per processor. It estimates how well-utilized the processors are in solving the problem, compared to how much effort is expended in communication and synchronization [2]. From its definition, it is clear that efficiency always stays between 0 and 1. Linear speedup corresponds to the highest efficiency $E_P = 1$. Efficiency close to 0 indicates that most effort of processors is wasted in communication and synchronization.

4.1.3 Scalability

Scalability of a parallel algorithm refers to its capacity to utilize more processors effectively. Later we will look closely at what problem parameters are playing a significant role in affecting the scalability of the Parareal Algorithm.

Definition 4.5. *(Scalability [23]) An algorithm is scalable if there is a minimal efficiency $\epsilon > 0$ such that given any problem of size N, there is a number of processors $P(N)$, which tends to infinity as N tends to infinity, such that the efficiency $E_{P(N)} \geq \epsilon > 0$ as N is made arbitrarily large.*

Scalability describes the performance of algorithms as the problem size is varied, in contrast to speedup which describes the performance of algorithms as the number of processors P varies. Scalability analysis provides a lower bound of the efficiency of the algorithm by choosing P dependening on the problem size [23].

Once again, two types of scalability, strong and weak, are of practical interest. In the case of weak scalability, the problem size is allowed to change as P is varied. On the other hand, in the case of strong scalability, all parameters used to specify the problem size are fixed as P is increased [23].

In general, scalability analysis is particularly useful for parallel algorithms which do not posses the property of linear speedup by design. It provides a method to find an optimal number of processors, if it exists, to maximize the performance of algorithms [23].

4.1.4 Theoretical Results

It is time to present some classical results about the performance of parallel algorithms. Two such results are **Amdahls law** and **Gustafsons law**.

Amdahl's law

There exist various statements of Amdahl's law and we select two most popular ones to present, one in terms of the expected speedup and the other in terms of the total parallel execution time. Both statements can be found in [23].

Let f denote the sequential fraction of the computation, and T_s be the execution time of a sequential run of the algorithm, then Amdahls law is stated as

Law 4.1. *(Amdahl's law v1 [4]) The speedup S_P, given P processors, is*

$$S_p = \frac{1}{f + (1-f)/p}. \tag{4.7}$$

Next, if we let T_{P_1} denote the execution time of the parallelized portion of the algorithm using a single processor and T_s denote the execution time of the sequential portion of the algorithm, then Amdahl's law can equivalently be stated as

Law 4.2. *(Amdahl's law v2 [4]) The wall clock time of the parallel execution, $T(P)$, of the algorithm given P processors is*

$$T(P) = T_s + \frac{T_{P_1}}{P} \tag{4.8}$$

Simple mathematical arguments on above definitions lead to the following useful insights of Amdahl's law.

Remark 4.1. *A perfectly-linear parallelizable algorithm is one in which f tends to 0, since $\lim_{f \to 0} S_P = P$.*

Remark 4.2. *The maximum speedup is limited by f^{-1}, since $\lim_{P \to \infty} S_P = \frac{1}{f}$.*

A good example representing Amdahl's law is borrowed from [58] and shown in Figure 4.1. It shows that the speedup of a parallel program is limited by its parallel portion. All speedup curves shown in the figure increase to their peaks and flatten out afterwards. Their peaks vary and highly depend on parallel portions of the program.

For instance, if 90% of the program can be parallelized, that is, 10% of the program must be executed sequentially, the maximum speedup using parallel computing would be 10x no matter how many processors are used [58].
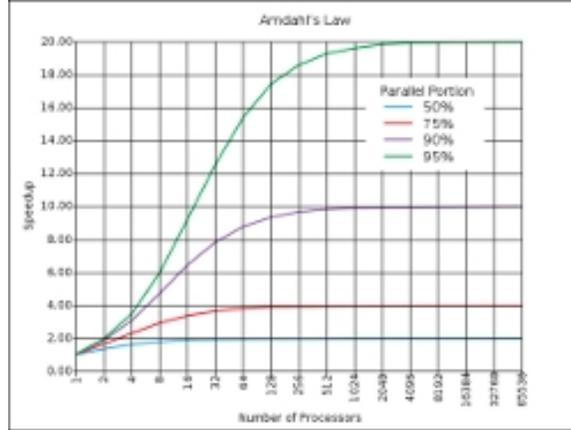
Figure 4.1: A graphical representation of Amdahl's law [58]

Gustafson's Law

Gustafson's Law (also known as Gustafson-Barsis' Law) is another law in computing, closely related to Amdahl's law.

Law 4.3. *(Gustafson's Law [22]) Gustafson's law defines the scaled speedup by keeping the parallel execution time constant and adjusting P as the problem size N changes*

$$S_{P,N} = P + \alpha(N) * (1 - P), \tag{4.9}$$

where $\alpha(N)$ is the non-parallelizable fraction of the normalized parallel time. Assuming that the serial function $\alpha(N)$ diminishes with the problem size N, then the speedup approaches P as N approaches infinity as desired in a linear scaling.

Gustafson's law reevaluates Amdahl's law in a primary aspect: Amdahl's law assumes the problem size is fixed and the sequential part of a program does not change with respect to the number of processors involved, whereas Gustafson's Law

49

has no such assumption, suggesting to adjust the problem size and use all available computing resources to solve the problem in given time.

Gustafson's law leads to a new path to select or reformulate problems so that solving a larger problem in the same amount of time would be possible. "In particular, the law redefines efficiency as a need to minimize the sequential part of a program, even if it increases the total amount of computation" [59].

Another example for Gustafson's Law is borrowed from [59] and shown in Figure 4.2. Differences between Amdahl's law and Gustafson's law can be clearly observed from Figure 4.1 and 4.2. Speedups in Figure 4.2 are straight lines with slopes $1 - \alpha(N)$, as shown in equation 4.9.
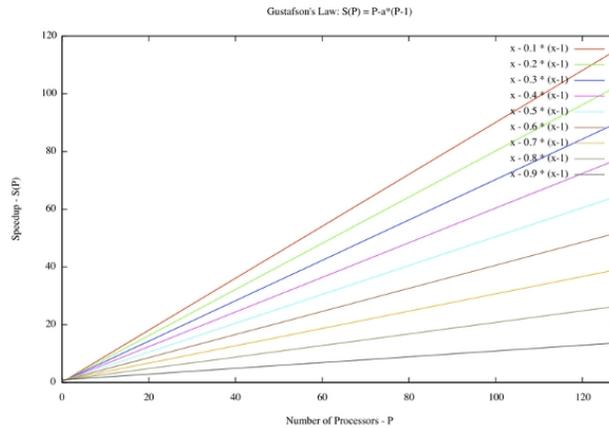
Figure 4.2: A graphical representation of Gustafson's Law [59]

4.1.5   Limitations of Classical Performance Analysis

It is pointed out in [23] that classical performance analysis has some practical problems, such as that a priori knowledge of the execution time of the sequential portion of the algorithm must be known. A new improved Karp-Flatt metric dealing with these issues was proposed by Alan Karp and Horace Flatt in 1990, and discussed in [23]. However, classical performance analysis is enough for this work, and we do not discuss the issue further since it is not the point of this work. Interested readers are referred to [23] for more details.

4.2 Space Parallelization (SP)

The most straightforward way to use multiple processing elements simultaneously to solve our problem is accomplished via *domain decomposition*, namely by breaking the spatial domain (the cable) into pieces so that each processor can execute its part of the algorithm on one piece simultaneously with the others. Since, by nature, the cable is thin and long, we only need to decompose it in one direction (x-direction).

This requires synchronization at each time step, which is possible only when all processors execute time steps of equal length. This is problematic for adaptive schemes, since local adaptivity may lead to unequal time steps among processors. However, in §4.4, we will describe a way to use even adaptive integrators in parallel within the Parareal Algorithm.

All *non-adaptive* numerical shemes have been parallelized for distributed computing via message passing, by (spatial) domain decomposition and implemented in modular code, written in C, using the MPI library for message passing.

We follow the *Master-Workers* paradigm. The full domain (entire cable) is divided evenly into N_w segments, with N_w the number of desired worker MPI processes (specified by the user at run-time). On a multiprocessor computer, each MPI process is executed on a separate processor. The master MPI process assigns each segment to one worker MPI process. At each time-step, worker processes exchange their boundary values with their adjacent neighbors; thus, all workers evolve at the same pace. The results are collected by the master process when the computations on all workers are finished.

Results of numerical experiments and comparison of solvers using space-only parallelization we have reported in [30] and are presented below.

Remark 4.3. *By requiring synchronization at every time step, i.e., updating and exchanging the boundary values at faces of control volumes before evolving to the next time step (for explicit schemes) or the next iteration (for implicit schemes), the numerical solution U_s obtained by serial implementation and the numerical solution*

U_p obtained by space parallized implementation of a certain time stepping scheme satisfy the relation

$$||V_s(x,t) - V_p(x,t)|| \sim eps, \ \text{for all } t \in (t_0, T),$$

where eps is the machine accuracy.

Remark 4.4. *As predicted by Amdahl's law, performance of parallelism by spatial domain decomposition deteriorates quickly for large number of processors. Therefore, there exists an optimal number of processors N_{opt}, depending on the selected solver, such that the performance of spatial parallelization is maximized.*

4.3 Time Parallelization - the Standard Parareal Algorithm (SPA)

After spatial parallelization, the bottleneck in time evolving problems is the time stepping. As the new generation of parallel computers/clusters provides more processors than can be filled up efficiently with space-only parallelization, it is natural to seek algorithms to parallelize in the time direction.

4.3.1 The Parareal Algorithm

The Parareal algorithm, first proposed by Lions, Maday, and Turinici in 2001 [31, 35], appears to be an effective algorithm for computing the numerical solution for general systems of ODEs of the form

$$u' = f(t,u), \ u(0) = u^0, \ t \in \Omega = (0,T). \tag{4.10}$$

where $f : R^M \longrightarrow R^M$ and $u : R \longrightarrow R^M$.

Similar to the idea of spatial domain decomposition, they introduced a decomposition of the temporal domain into time segments. Also, in the spirit of predictor-corrector schemes, they introduced both coarse and fine time integrators. Their solutions are then combined in a corrector scheme which allows for the coarse solution to be updated iteratively while preserving the accuracy and stability of the time discretization. The coarse time integrator approximation and the application of the corrector scheme are purely serial in the implementation. The fine approximations are serial only within each time segment, thus allowing for parallel execution of the fine solver on each of these time segments. The corrector scheme is then used to update the coarse solver approximation using the results of the fine approximations on each segment, which have been computed concurrently (in parallel), and this procedure is iterated until convergence.

The Parareal Algorithm is completely problem independent and leaves much flexibility in the choice of time discretization and solver. It is important to note that the algorithm can never exceed the accuracy or stability of the numerical schemes being employed. Also, when we talk about the convergence of the algorithm, we do so in terms of its approach towards the solution that would have been obtained if the problem was solved directly using the fine solver over the entire temporal domain. Another aspect of this algorithm, which makes it especially promising for real time computations, and in stark contrast to the more traditional spatial decompositions, is that in a true parallel implementation the algorithm requires a very minimal amount of communication between any of the processors carrying out the fine approximations.

A brief description of the *Standard Parareal Algorithm* (SPA) is as follows. We divide the time interval $\Omega_T = [0, T]$ into N time segments $\Omega_n = (t_n, t_{n+1}), n = 0, 1, ..., N-1$, with $0 = t_0 < t_1 < ... < t_{N-1} < t_N = T$ and $\Delta t_n := t_{n+1} - t_n$. Two propagation operators, G and F, are required. The coarse operator $G(t_2; t_1, u_1)$ is implemented sequentially to provide a rough approximate solution U_2 of equation (4.10) at time t_2 with initial condition $U_1 = u(t_1)$ at time t_1. On the other hand, the fine operator $F(t_2; t_1, u_1)$ is implemented in parallel within each segment Ω_n to

improve the accuracy. The algorithm starts with initial approximations $U_n^0, n = 1, 2, ..., N$, at times $t_1, t_2, ..., t_N$ obtained from the sequential computation of $U_{n+1}^0 = G(t_{n+1}; t_n, U_n^0)$, with given initial condition $U_0^0 = u(t_0)$, and then performs for $k = 0, 1, 2, ...$ the correction iteration

$$U_{n+1}^{k+1} = G(t_{n+1}; t_n, U_n^{k+1}) + F(t_{n+1}; t_n, U_n^k) - G(t_{n+1}; t_n, U_n^k), \qquad (4.11)$$

until a pre-specified tolerance is satisfied or the maximum number of iterations is reached. A pseudo-code implementing SPA can be found in [50] and is shown in Table 4.1 below.

Table 4.1: Standard Parareal Algorithm [50]

$\lambda_0^0 \leftarrow u_0$
for $i = 0 : N - 1$ **do**
    $\lambda_{i+1}^0 \leftarrow G(\lambda_{i+1}^0; t_i, \lambda_i^0)$
end for
solve $F(\lambda_{i+1}^0; t_i, \lambda_i^0)$ concurrently on $i = 0, ..., N - 1$ processors with one fine subproblem per processor.
$k \leftarrow 0$
while true **do**
    $\lambda_0^{k+1} \leftarrow \lambda_0^k$
    for $i = 0 : N - 1$ **do**
        solve $G(\lambda_{i+1}^{k+1}; t_i, \lambda_i^{k+1})$
        $\lambda_{i+1}^{k+1} \leftarrow G(\lambda_{i+1}^{k+1}; t_i, \lambda_i^{k+1}) + F(\lambda_{i+1}^k; t_i, \lambda_i^k) - G(\lambda_{i+1}^k; t_i, \lambda_i^k)$
    end for
    if convergence **then**
        break
    end if
    solve $F(\lambda_{i+1}^{k+1}; t_i, \lambda_i^k)$ in parallel on $i = 0, ..., N - 1$ processors with one fine subproblem per processor.
    $k \leftarrow k + 1$
end while

4.3.2 Properties of the Parareal Algorithm

For easy reference, we list here as remarks the main properties of SPA, already mentioned in the previous section.

Remark 4.5. *In contrast to more traditional spatial decomposition methods, described in §4.2, the Parareal Algorithm is a pure parallel algorithm which requires no communication between processors carrying out the fine propagator F [50, 23].*

Remark 4.6. *Provided G and F are convergent and stable for their own chosen timesteps ΔT and δt (usually, $\Delta T >> \delta t$), then, for iteration k (with $k = 0$ being the first iteration)*

$$||U_s - U_{sp}|| \sim eps, \quad t \in (t_0, k\Delta T),$$

where U_s is the numerical solution obtained by serial F, U_{sp} is the numerical solution obtained by the SPA, and eps is the machine accuracy. This means that

$$||U_s - U_{sp}|| \sim eps, \quad t \in (t_0, T),$$

at $N - 1 = \frac{T-t_0}{\Delta T} - 1$ iterations. Moreover, at ieration $k \leq N - 1$,

$$||U_s - U_{sp}|| \sim eps, \quad t \in (t_0, t_k).$$

Basically, it means that the iterative correction formula (4.11) converges towards the serial solution obtained by using the same fine propagator F on the same discrete grids in time (and space). A series of intermediate values U_n satisfies $U_{n+1} = F(t_{n+1}; t_n, U_n)$ in finite number of iterations. That is, the approximation at the time point t_n will have achieved the accuracy of the F-propogator [50].

Remark 4.7. *SPA is independent of the choice of propagators G and F. The only requirement is that the chosen methods be convergent and stable for the specified timesteps ΔT and δt [50].*

4.3.3 Convergence, Stability and Performance of SPA

Convergence and stablility as well as performance analysis of SPA are important to us. A nice general overview of all of the basic mathematical results is provided in [50]. Interested readers are refferred to [6, 18, 17, 34, 51] for a thorough analysis of mathematical results on convergence and stability.

Convergence Results

Two primary convergence results were published in [17, 34]. We will state both of them, starting with the one provided by Gander and Hairer in [17].

Assuming all the time segments are of the same size, $\Delta t_n = \Delta t := T/N, n = 0, 1, ..., N-1$, and that F is the exact solution, i.e., $F(t_n; t_{n-1}, U_{n-1}^k) = \varphi \Delta t_{n-1}(U_{n-1}^k)$, the evaluation problem on each time segment is considered

$$u_n^{'}(t) = f(t, u_n(t)),\ t \in (t_n,\ t_{n+1}),\ u_n(t_n) = U_n,\ n = 0, 1, ..., N-1,$$

where the initial values U_n on the time segment Ω_n coincides with the solution of (4.10) on Ω_n, i.e., U_n satisfies the system of equations

$$U_0 = u^0,\ U_n = \varphi_{\Delta t_{n-1}}(U_{n-1}),\ n = 0, 1, ..., N-1,$$

where $\varphi_{\Delta t_{n-1}}$ denotes the solution of (4.10) with initial condition U after time Δt_n.

Another assumption is that the difference between the approximate solution obtained by G and the exact solution can be expanded for Δt small

$$F(t_n; t_{n-1}, x) - G(t_n; t_{n-1}, x) = c_{p+1}(x)\Delta t^{p+1} + c_{p+2}(x)\Delta t^{p+2} + ... \qquad (4.12)$$

Expansion (4.12) can be achieved if, for example, the right hand side function f is smooth enough, and G is a Runge-Kutta method.

With the final assumption that G satisfies the Lipschitz condition

$$||G(t + \Delta t; t, x) - G(t + \Delta t; t, y)|| \leq (1 + C_2 \Delta t)||x - y||, \qquad (4.13)$$

the following convergence theorem is proved in [17].

Theorem 4.1. *(Convergence, Gander and Hairer [17])*
Let $F(t_n; t_{n-1}, U_{n-1}^k) = \varphi_{\Delta t_{n-1}}(U_{n-1}^k)$ be the exact solution on time segment Ω_{n-1}, and let $G(t_n; t_{n-1}, U_{n-1}^k)$ be an approximate solution with local truncation error bounded by $C_3 \Delta t^{p+1}$, and statisfying (4.12), where the $c_j, j = p+1, p+2, ...$ are continuously differentiable, and assume that G satisfies the Lipschitz condition (4.13). Then, at iteration k of the Parareal Algorithm (4.10), we have the bound

$$
\begin{aligned}
||u(t_n) - U_n^k|| &\leq \frac{C_3}{C_1} \frac{(C_1 \Delta t^{p+1})^{k+1}}{(k+1)!} (1 + C_2 \Delta t)^{n-k-1} \prod_{j=0}^{k}(n-j) \\
&\leq \frac{C_3}{C_1} \frac{(C_1 t_n)^{k+1}}{(k+1)!} e^{C_2(t_n - t_{k+1})} \Delta t^{p(k+1)}. \qquad (4.14)
\end{aligned}
$$

A similar convergence result is obtained by Maday, et.al [34] by introducing the propagator ε (4.13) defined by $u(t) = \varepsilon_{t-t_0}(t_0, u_o) = \varepsilon_{t-\tau}(\tau, u(\tau))$ for any $\tau \geq t_0$. The existence of such a propagator follows, e.g. from the hypothesis on f:

$$||f(t, x)|| \leq C(1 + ||x||), \quad ||f(t, x) - f(t, y)|| \leq C||x - y||,$$

and, in addition, the following stability result:

$$||\varepsilon_\tau(t, x) - \varepsilon_\tau(t, y)|| \leq (1 + C\tau)||x - y||. \qquad (4.15)$$

With assumptions that propagator G and F satisfy the semi-group property

$$G_{t-t_0}(t_0, u_0) = G_{t-\tau}(\tau, G_{\tau-t_0}(t_0, u_0)), \quad F_{t-t_0}(t_0, u_0) = F_{t-\tau}(\tau, F_{\tau-t_0}(t_0, u_0)),$$

and, for any $\tau \geq t_0$

$$||\delta F_\tau(t,x)|| \leq C\tau\eta(1+||x||), \ ||\delta G_\tau(t,x)|| \leq C\tau\epsilon(1+||x||), \qquad (4.16)$$

where δF and δG are the differences $\delta F = \varepsilon - F$ and $\delta G = \varepsilon - G$, respectively, one can show from (4.15) - (4.16) that

$$||F_\tau(t,x) - F_\tau(t,y)|| \leq (1+C\tau)||x-y||$$

and

$$||G_\tau(t,x) - G_\tau(t,y)|| \leq (1+C\tau)||x-y||,$$

with a constant, still denoted by C, as any constant that does not depend on Δt, nor τ, η or ϵ.

Eventually, the following theorem is obtained under these assumptions by Mayday, Ronquist and Staff in [34].

Theorem 4.2. *(Convergence, Maday, Ronquist, and Staff [34])*

Assume that the discrete propagators F and G satisfy (4.15) and (4.16). Assume also that $k \leq K$ with some fixed $K \leq N/2$ and that we have a constant C dependent of u_0, T and k. The error between the exact solution and the solution provided by the Parareal Algorithm (4.14) satisfies

$$||U_n^k - u(t_n)|| \leq C(\epsilon^k + \eta), \ \forall t_n < T, \qquad (4.17)$$

Stability Results

Stability is studied by Bar, Ronquist and Staff in [6, 51]. Stability results for an autonomous differential equation are obtained by Staff and Ronquist in [51]. More general discussion can be found in [6].

Theorem 4.3. *(**Stability, Staff and Ronquist [51]**)*

Assume we want to solve the autonomous differential equation

$$y' = \mu y, \; y(0) = y_0, \; 0 > \mu \in R,$$

and that $-1 \leq r, R \leq 1$ *where* $r = r(\mu \delta t)$ *is the stability function for the fine propagator* F *using time step* δt *and* $R = R(\mu \Delta T)$ *is the stability function for the coarse propagator* G *using time step* ΔT. *Then the Parareal Algorithm is stable for any number of segments* N *and any number of iterations* $k \leq N$ *as long as*

$$\frac{\overline{r} - 1}{2} \leq R \leq \frac{\overline{r} + 1}{2}, \tag{4.18}$$

where $\overline{r} = r(\mu \delta t)^s$ *and* $s = \Delta T / \delta t$.

It is still not obvious from (4.18) which solvers will fulfill this stability condition. The next theorem gives some insight by considering a special case [51].

Theorem 4.4. *(**Stability, Staff and Ronquist [51]**)*

Assume we want to solve the autonomous differential equation

$$y' = \mu y, \; y(0) = y_0, \; 0 > \mu \in R,$$

using the Parareal Algorithm. Assume also that the system is stiff, meaning that $z = \mu \Delta T << -1$, *and that the fine propagator is close to exact. Then the "stability function" can be written as*

$$H(n, k, R) = (-1)^k \begin{pmatrix} n - 1 \\ k \end{pmatrix} R^n,$$

and stability is guaranteed if the following property is fulfilled:

$$R_\infty = \lim_{z \to -\infty} |R(z)| \leq \frac{1}{2}. \tag{4.19}$$

Performance Analysis

Calculation of the speedup and efficiency of the Parareal algorithm, in both cases of strong and weak scalings, is standard and straightforward. Our results agree with those reported by Samaddar, Newman and Sanchez in [45]. To be consistent, we borrow their notations here.

It is worth pointing out that both implementations share the same idea attempting to incorporate space-parallelization into the framework of the Parareal Algorithm to accelerate computations. However, their implementation is quite different from ours in some major aspects.

First, their implementation is via scripts external to the code used to solve a problem, whereas ours is built within the code. Second, one important feature of their implementation is to break a large MPI job to many small ones running concurrently, each demanding only small amount of processors. This feature is useful, for example, when running the MPI job on a cluster accessible to many users. However, intermediate values, produced by previous MPI jobs, must be saved onto disk and loaded later into memory by future MPI jobs as starting values, thus incurring performance penalties. Our implementation does not have this feature; it allocates all processors at the moment the MPI job is submitted; thus no additional I/O operations between memory and hard-disk are required.

We first consider the case of strong scaling, in which the problem is solved up to fixed final time T. Let $T_G^{ser}(T)$ and $T_F^{ser}(T)$ be the wallclock times to solve the problem serially up to time T using propagators G and F, respectively. Define the parameter β as the ratio of these two times,

$$\beta = \frac{T_F^{ser}(T)}{T_G^{ser}(T)}, \tag{4.20}$$

Clearly $\beta > 1$ since G is the cheaper and faster propagator.

Let N be the total number of processors. It is the number of time segments as well, since the fine propagator F, running in parallel, requires one processor per time

segment of length $\Delta T = T/N$, assuming the whole simulation time perild Ω_T is divided evenly into segments of the same length.

We denote by $k_s(N)$, a function of N, the number of iterations the Parareal takes until converges. It is not hard to see that the total consumed time to solve the problem is

$$T_{sp}^s = K_s(N) \left(T_G^{ser}(T) + \frac{T_F^{ser}(T)}{N} \right). \tag{4.21}$$

From equation 4.21, we obtain an estimate of the speedup and efficiency in the case of strong scaling.

Proposition 4.1. (*Speedup and efficiency for strong scaling*)

The strong scaling parallel speedup factor (or gain) for the Parareal Algorithm is given by:

$$S_{sp}^s = \frac{T_F^{ser}(T)}{T_{sp}^s} = \left[k_s(N) \left(\frac{1}{\beta} + \frac{1}{N} \right) \right]^{-1}, \tag{4.22}$$

and the associated efficiency is

$$E_{sp}^s = \frac{S_{sp}^s}{N} = \left[k_s(N) \left(\frac{N}{\beta} + 1 \right) \right]^{-1}. \tag{4.23}$$

Note that the typical strong scaling for spatial parallelization, $S_p(N) = N$ is only recovered when $\beta \to \infty$ and $k_s(N) = 1$. But in the parareal case, $k_s(N)$ is a function of N and β is finite. Efficiency of the parareal will depend on the value $k_s(N)$, which depends on the choice of the coarse solver G. But even without knowing $k_s(N)$ at this point, several things can be learnt from this model. First, β seems to roughly set the maximum number of processors for which SPA yields any net parallel gain. For N much larger than this value, the serial part of the algorithm dominates the calculation and, as predicted by Amdahls law, performance deteriorates quickly. Secondly, a net parallel gain is obtained only for as long as $k_s(N) < N$.

Proposition 4.2. *(**Maximum speedup**)*

The speedup, assuming $k \geq 1$, is maximized for $k = 2$ and $\Delta T = \sqrt{2T\delta t}$. From this it can be dedudcd that

$$S^s_{sp} = \frac{1}{2}\sqrt{T}2\delta t, \quad N = \sqrt{T}2\delta t, \quad E^s_{sp} = \frac{1}{2}. \tag{4.24}$$

Remark 4.8. *Equ.(4.24) seems to say that the efficiency is bounded by the factor $\frac{1}{2}$. But Bal [5] shows that this can be overcome by introducing a multi-step system. We assume that we have a scale of time steps such that*

$$\Delta_m T \ll \Delta_{m-1} T \ll ... \ll \Delta_1 T \ll \Delta_0 T \ll T.$$

The speeup of the multi-step algorithm is then given by

$$S = \frac{\frac{T}{\Delta_m T}}{N \left(2 \left(\frac{\Delta_0 T}{\Delta_1 T} + ... + \frac{\Delta_{m-1} T}{\Delta_m T} \right) + \frac{\Delta_{m-1} T}{\Delta_m T} \right)}. \tag{4.25}$$

By using this multi-step method, which is implemented as a restarted algorithm (see [5] for details), the efficiency can be close to 1.

Weak Scaling Study

We next consider the case in which the problem is to be solved up to time $T = N\Delta T$, with ΔT fixed. That is, the length of time (hence the problem size) increases linearly with the number of processors. As mentioned previously, in perfect parallelism one would expect the wallclock time to be independent of N, since each processor would be doing exactly the same amount of work. That is,

$$W = \frac{T_N(N * \Delta T)}{T_1(\Delta T)} = 1, \tag{4.26}$$

where $T_n(t)$ denotes the wallclock time needed to solve a problem of length t using n processors. In the parareal case, the time needed to solve a problem of size $N\Delta T$

using N processors is:

$$T^w_{sp} = k_w(N)(N \cdot T^{ser}_G(\Delta T) + T^{ser}_F(\Delta T)), \tag{4.27}$$

from which the work per processor becomes:

$$W^w_{sp}(N) = \frac{T^w_{sp}}{T^{ser}_F(\Delta T)} = k_w(N) \left(1 + \frac{N}{\beta} \right). \tag{4.28}$$

Proposition 4.3. *(**Speedup and efficiency for weak scaling**)*

The weak scaling parallel speed-up factor (or gain) for the Parareal Algorithm is given by:

$$S^w_{sp} = \frac{T^{ser}_F(N\Delta T)}{T^w_{sp}} = \left[k_w(N) \left(\beta + \frac{1}{N} \right) \right]^{-1}, \tag{4.29}$$

and the coresponding efficiency is

$$E^w_{sp} = \frac{S^w_{sp}}{N} = [k_w(N)(N\beta + 1)]^{-1}. \tag{4.30}$$

Note that the function $k_w(N)$ is different from $k_s(N)$, since now T is not kept fixed. As before, the classical weak scaling for the spatial case, $W_{sp} = 1$, is only recovered if $\beta \to \infty$ and if $k_w(N) = 1$. This will certainly not be the case. Again, it seems that β roughly sets the maximum number of processors for which a favorable scaling for the work-per-processor should be expected, although how good the scaling would be ultimately depends on the form of $k_w(N)$.

4.4 Time-and-space Parallelization - the Extended Parareal Algorithm (EPA)

Noticing that our space-parallized solvers are essentially capable of reproducing the numerical solutions obtained by their serial version (Remark 4.3), and that we have freedom to choose which solvers to use for G and F in SPA (Remark 4.7), it is natural

to think of incorprating the space-parallelized solvers into the framework of SPA to achieve both time and space parallelization.

However, the naive thought of substituting space-parallelized ones for the serial propagators G and F of SPA is not optimal, in the sense that more processors waste their power to unnecessarily repeat previous work due to the fact that in SPA all iterations always start from the first time segment.

We hereby propose an extension to the Parareal Algorithm (described in Table 4.2), to effectively bring the space-parallelized solvers into the framework of SPA. The input parameters of the extended algorithm are listed in Table 4.3.

Again, the code implementing the Extended Parareal Algorithm is written in modular C, so that other Hodgkin-Huxley type of ionic models and/or other time integrators can be incorporated easily, and it uses the MPI library for parallization. The code follows the Master-Slave structure and object-oriented design in which the master process is not involved in the computation.

We see that there are dozens of input parameters involved, which give us great flexibility for various simulation purposes. We list some important setups to show that the algorithm proposed in Table 4.2, and its implementaion, is able to not only incorprate space parallized solvers, but also to reproduce the simpler algorithms described in previous sections. This makes it possible to compare performance of the schemes from within the same code.

Remark 4.9. *(Special Setup I: reproduction of the serial implementations)*
The numerical solution obtained by the coarse propagator G (Table 4.2) reproduces the numerical solution obtained by the serial implementation (§3.5). This is achieved simply by setting $nWRs = 1$, $nTS = 1$, $CSP = 0$, and $FTS = none$.

Table 4.2: Extended Parareal Algorithm

Divide the time interval $[0, T]$ into N time segments $[t_n, t_{n+1}]$, $0 = t_0 < ... < t_N = T$. Assume M processors at disposal for parallel computing, where $M \geq N$.

$U_0^{\text{old}} \leftarrow u_0$
for $n = 0 : N - 1$ **do**
    Apply space-parallelized coarse operator G, using $\min\{M, n_G\}$ processors,
    on time segment $[t_n, t_{n+1}]$ with initial value U_n^{old}.
    $U_{n+1}^{\text{old}} \leftarrow G(t_{n+1}; t_n, U_n^{\text{old}})$
end for
if no fine operator F is defined **then**
    print U_n^{old} for $n = 0, ..., N$ and return
end if
$U_0^{\text{new}} \leftarrow u_0$, $G_n^{\text{old}} \leftarrow U_n^{\text{old}}$ for $n = 0, ..., N$, $G_0^{\text{new}} \leftarrow U_0^{\text{old}}$
$k \leftarrow 0$
while true **do**
    Apply space-parallelized fine operator F, using $\min\{\frac{M}{N-k}, n_F\}$ processors per
    time segment, concurrently on time segments $[t_n, t_{n+1}]$ with intial values
    U_n^{old}, $n = k, ..., N - 1$.
    $U_{n+1}^{\text{new}} \leftarrow F(t_{n+1}; t_n, U_n^{\text{old}})$ for $n = k, ..., N - 1$
    $k \leftarrow k + 1$
    if $k = N$ **then**
        print U_n^{new} for $n = 0, ..., N$ and break
    end if
    for $n = k : N - 1$ **do**
        Apply space-parallelized coarse operator G, using $\min\{M, n_G\}$ processors, on
        time segment $[t_n, t_{n+1}]$ with initial value U_n^{new}.
        $G_{n+1}^{\text{new}} \leftarrow G(t_{n+1}; t_n, U_n^{\text{new}})$
        $U_{n+1}^{\text{new}} \leftarrow G_{n+1}^{\text{new}} + U_{n+1}^{\text{new}} - G_{n+1}^{\text{old}}$
    end for
    if $\max_{k \leq n \leq N} |U_n^{\text{new}} - U_n^{\text{old}}| \leq \text{TOL}$ **then**
        print U_n^{new} for $n = 0, ..., N$ and break
    end if
    $U_n^{\text{old}} \leftarrow U_n^{\text{new}}$ for $n = k, ..., N$
    swap G^{old} and G^{new}
end while

Table 4.3: Input Parameters for the Extended Parareal Algorithm

Symbol	Unit	Meaning
nWRs	—	number of MPI worker processors
tstart	ms	starting time
tmax	ms	maximum simulation time
dthist	ms	frequency to print out history
dtprof	ms	frequency to print out profile
nTS	—	number of time segments
TOL	—	tolerance to stop iteration of parareal algorithm
CTS	—	choice of coarse propagator G
CSP	—	type of coarse propagator G. (0=serial, 1=parallel)
cWRs	—	($= n_G$), optimal number of processors for propagator G
cNsub	—	number of substeps if propagator G = STS
cdamp	—	damping number if propagator G = STS
FTS	—	choice of fine propagator F
FSP	—	type of propagator F. (0 = serial, 1 = parallel)
fWRs	—	($= n_F$), optimal number of processors for propagator F
fNsub	—	number of substeps if propagator F = STS
fdamp	—	damping number if propagator F = STS
dtfacG	—	rescaling factor for time step in propagator G
dtfacF	—	rescaling factor for time step in propagator F
MM	—	number of control volumes in the smallest biological cell
minCell	μm	minimal biological cell length
maxCell	μm	maximal biological cell length
lend	μm	left end of cable
rend	μm	right end of cable
APD[0]	—	index of first node used for computing APD
APD[1]	—	index of second node used for computing APD
SMODEL	—	choice of ionic model (1 = HH, 2 = LR)
Cm	$\mu F/cm^2$	membrane capacitance
Ri	$\Omega\,cm$	resistivity inside biological cells
R$_{\text{gap}}$	$\Omega\,cm$	gap junction resistivity between cells
a	μm	radius of the cable

Symbol	Unit	Meaning
KOnormal	mM	normal value of $[K]_o$
U$_0$	—	array of initial values in normal region of cable
KOabnorm	mM	abnormal value of $[K]_o$
U$_1$	—	array of initial values in abnormal region of cable
stim_period	ms	stimulus peroid
stim_dur	ms	stimulus duration
stim_start	ms	stimulus starting time
stim_end	ms	stimulus end time
stim_range	μm	stimulus range $[0, range]$
stim_ampl	$\mu A/cm^2$	stimulus amplitude

Remark 4.10. (*Special Setup II: reproduction of the space parallized implementations*)

The numerical solution obtained by the coarse propagator G of EPA reproduces the numerical solution obtained by the space parallized implementation (§4.2). This is achieved by setting $nWRs = 1$, $nTS = 1$, $CSP = 1$, and $FTS = none$.

Remark 4.11. (*Special Setup III: reproduction of SPA implementations*)

The numerical solution obtained by the Extended Parareal Algorithm reproduces the numerical solution obtained by the Standard Parareal implementation. This is achieved by setting $CSP = 0$, and $FSP = 0$.

We list some other important features of EPA here.

Remark 4.12. *By numerical experiments we determine the parameters n_G and n_F so that the coarse propagator G and the fine propagator F each achieves its best performance when these numbers of processors are used. These numbers are used whenever the available processors are more than these numbers in order to avoid unnecssray communication overheads in practice.*

Remark 4.13. *There are numerous combinations of coarse and fine propagators (lower order & higher order, larger time-step & smaller time-step, etc).*

Remark 4.14. *The extended Parareal algorithm inherits numerical properties, such as stability and convergence, from the Standard Parareal Algorithm but it has very different performance. When the coarse and fine propagators are both spatially parallized, the number of processors used for the coarse propagator G per subproblem is*

$$\Delta n_G = \begin{cases} n_G, & \text{if } M \geq n_G \\ \\ M, & \text{otherwise} \end{cases} = \min\{M, n_G\},$$

and the number of processors used for the fine propagator F per subproblem is

$$\Delta n_F^N = \begin{cases} n_F, & \text{if } \frac{M}{N} \geq n_F \\ \\ \frac{M}{N}, & \text{otherwise} \end{cases} = \min\left\{\frac{M}{N}, n_F\right\},$$

where M is the total number of worker processors, N is the number of time segments such that $N \leq M$, and k is the count of parareal iterations.

Performance Analysis

Let the time interval $[0, T]$ of the simulation be divided into N segments each of lenght $\Delta T = T/N$. Denote the wallclock time to solve the problem using space parallized coarse solver G with n processors by $T_G^n(T)$, and to solve it using space parallized fine solver F with n processors be $T_F^n(T)$. Note that *the fine solver F is not only parallized in space, but also parallized in time*. We denote the wallclock time to solve the problem using time-and-space parallized F with n processors by $\widetilde{T}_F^n(T)$. Finally, we assume that T_G^n and T_F^n satisfy the following relations,

$$T_G^n(\Delta T) = \begin{cases} \frac{1}{n}T_G^n(n * \Delta T) = \frac{1}{n}T_G^{ser}(\Delta T), & n \leq n_G \\ \\ T_G^{n_G}, & n > n_G \end{cases}$$

$$T_F^n(\Delta T) = \begin{cases} \frac{1}{n}T_F^n(n * \Delta T) = \frac{1}{n}T_F^{ser}(\Delta T), & n \leq n_F \\ \\ T_F^{n_F}, & n > n_F \end{cases} \tag{4.31}$$

68

for all $\Delta T \leq T$. In other words, we assume that the spatially parallelized G and F maintain linear speedup until they reach their speedup limits at n_G and n_F, respectively.

Now we are in position to disscuss the performance of EPA. Let M be the total number of worker processors such that $M > N$. Without counting communication overhead caused by the space and time parallizations, the fact that the coarse propagator G runs sequentially in time yields

$$T_G^M(\Delta T) = T_G^{\Delta n_G}(\Delta T) = \frac{1}{N}T_G^{\Delta n_G}(T) = \frac{1}{N}T_G^M(T) = \frac{1}{N\Delta n_G}T_G^{ser}(T)$$

$$\implies T_G^M(T) = \frac{1}{\Delta n_G}T_G^{ser}(T),\ T_G^M(\Delta T) = \frac{1}{N\Delta n_G}T_G^{ser}(T)$$

$$\implies
\begin{cases}
T_G^M(T) = \frac{1}{M}T_G^{ser}(T),\ T_G^M(\Delta T) = \frac{1}{NM}T_G^{ser}(T), & M \leq n_G \\
\\
T_G^M(T) = \frac{1}{n_G}T_G^{ser}(T),\ T_G^M(\Delta T) = \frac{1}{Nn_G}T_G^{ser}(T), & M > n_G
\end{cases}
\tag{4.32}$$

On the other hand, the fact that the fine propagator F runs in parallel in time suggests

$$\widetilde{T}_F^M(T) = T_F^{\Delta n_F^N}(\Delta T) = \frac{1}{\Delta n_F^N}T_F^{ser}(\Delta T) = \frac{1}{N\Delta n_F^N}T_F^{ser}(T)$$

$$\implies \widetilde{T}_F^M(T) = \frac{1}{N\Delta n_F^N}T_F^{ser}(T),\ T_F^{\Delta n_F^N}(\Delta T) = \frac{1}{N\Delta n_F^N}T_F^{ser}(T)$$

$$\implies
\begin{cases}
\widetilde{T}_F^M(T) = \frac{1}{M}T_F^{ser}(T),\ T_F^M(\Delta T) = \frac{1}{M}T_F^{ser}(T), & M \leq N*n_F \\
\\
\widetilde{T}_F^M(T) = \frac{1}{Nn_F}T_F^{ser}(T),\ T_F^{\Delta n_F}(\Delta T) = \frac{1}{Nn_F}T_F^{ser}(T), & M > N*n_F
\end{cases}
\tag{4.33}$$

If we take $G = F$, and therefore $T_G^{ser} = T_F^{ser}$ and $n_G = n_F$, then equation (4.32) represents the performance of space parallelized F and equation (4.33) represents the performance of time-and-space parallelized F running on the same number of processors M to an identical time T. We obtain some interesting but very important observations on the performance of EPA by comparing (4.32) and (4.33).

69

First, when we have limited number of processors, such as $M \leq n_F$, the overall speedups obtained by space parallized F and time-and-space parallized F are the same, $T_F^{ser}/T_F^M = T_F^{ser}/\widetilde{T}_F^M = M$, even though they have different speedups on segments ΔT. It indicates that, in this case, EPA performs worse than SP since additional computational cost of the coarse propagator is invovled in each iteration, plus that usually it takes more than one iteration to converge, and therefore involves more computational cost of the fine propagator. If more processors come into play, such as $M > n_F$, then SP cannot take any advantage of additional processors, but EPA can, and now EPA begins to show its merit over SP, since $\widetilde{T}_F^M/T_F^M = n_F/M < 1$. The more processors we have available, the better performance EPA achieves in one iteration, and if the computational cost of the coarse propagator is far less than that of the fine propagator, EPA could beat SP provided it converges after not too many iterations. In the extreme case, $M \geq N * n_F$, EPA achieves its best, N times faster than SP, which is linearly proportional to the number of time segments N.

Next, notice that T_F^M for space parallelized F is independent of the number of time segments N, as shown in (4.32), but $\widetilde{T}_F^M$ for EPA, shown in (4.33), is inversely proportional to N. This indicates that, in the case of strong scaling, when simulation time T is fixed, we can break T into more segments with fixed number of processors per time segments to achieve better speedup when more and more processors are available. On the other hand, in the case of weak scaling, when ΔT and number of processors n_F used on each segment are fixed with $T = N * \Delta T$ varying, then $\widetilde{T}_F^M$ stays the same, while T_F^M increases linearly with N. This again makes EPA N times faster than SP. So, in both scaling senses, EPA has the potential to perform better than SP when more and more processors are at our disposal.

Although the above discussion is rough, it gives us a sound direction to think of when EPA is useful. Now we need to carefully calculate the computational cost T_{ep}^s of EPA, converging in k_s iterations, with $k_s \leq N$, to have a better insight of this

algorithm. T_{ep}^s can be estimated (ignoring overheads) as

$$
\begin{aligned}
T_{ep}^s &= \underbrace{T_G^M(T) + \widetilde{T}_F^M(T)}_{\text{1st iteration}} + \underbrace{T_G^M(T - \Delta T) + \widetilde{T}_F^M(T - \Delta T)}_{\text{2nd iteration}} + \cdots \\
&\quad + \underbrace{T_G^M(T - (k_s - 1)\Delta T) + \widetilde{T}_F^M(T - (k_s - 1)\Delta T)}_{k_s \text{ iteration}} \\
&= \left(T_G^M(T) + \cdots + T_G^M(T - (k_s - 1)\Delta T)\right) + \left(\widetilde{T}_F^M(T) + \cdots + \widetilde{T}_F^M(T - (k_s - 1)\Delta T)\right) \\
&= (N + \cdots + (N - k_s + 1))T_G^{\Delta n_G}(\Delta T) + \left(T_F^{\Delta n_F^N}(\Delta T) + \cdots + T_F^{\Delta n_F^{N-k_s+1}}(\Delta T)\right) \\
&= \frac{(2N - k_s + 1)k_s}{2}T_G^{\Delta n_G}(\Delta T) + \left(T_F^{\Delta n_F^N}(\Delta T) + \cdots + T_F^{\Delta n_F^{N-k_s+1}}(\Delta T)\right). \tag{4.34}
\end{aligned}
$$

Case I: When $M < \min\{n_G, n_F\}$, EPA is slower than SP for any $r_s \geq 1$, since

$$
T_{ep}^s > T_F^{\Delta n_F^N}(\Delta T) = T_F^{M/N}(\Delta T) = \widetilde{T}_F^M(T) = \frac{1}{M}T_F^{ser} = T_F^M(T).
$$

Case II: When $M \geq \max\{n_G, N * n_F\}$, without loss of generality, we assume $n_G < N * n_F$ and $M = N * n_F$ and have

$$
\Delta n_G = n_G, \text{ and } \Delta n_F^n = n_F, \text{ for all } n = N, ..., N - k_s + 1,
$$

thus,

$$
\begin{aligned}
T_{ep}^s &= \frac{(2N - k_s + 1)k_s}{2}T_G^{n_G}(\Delta T) + k_s * T_F^{n_F}(\Delta T) \\
&= \frac{(2N - k_s + 1)k_s}{2Nn_G}T_G^{ser}(T) + \frac{k_s}{Nn_F}T_F^{ser}(T), \tag{4.35}
\end{aligned}
$$

and the speedup and efficiency are given by

$$
S_{ep}^s = \frac{T_F^{ser}}{T_{ep}^s} = \frac{N}{\frac{(2N-k_s+1)k_s}{2n_G\beta} + \frac{k_s}{n_F}}, \tag{4.36}
$$

and

$$E_{ep}^s = \frac{S_{ep}^s}{M} = \frac{N}{n_F} \frac{1}{\frac{(2N-k_s+1)k_s}{2n_G\beta} + \frac{k_s}{n_F}} = \frac{N}{\frac{(2N-k_s+1)k_s\alpha}{2\beta} + k_s}. \tag{4.37}$$

where $\alpha = n_F/n_G$.

To achieve better performance than space parallized F, we want

$$\frac{T_F^M}{T_{ep}^s} = \frac{T_F^{n_F}}{T_{ep}^s} = \frac{1}{n_F}\frac{T_F^{ser}}{T_{ep}^s} = \frac{S_{ep}^s}{n_F} = E_{ep}^s > 1.$$

That is, we want

$$\frac{\alpha}{2\beta}k_s{}^2 - \left(\frac{(2N+1)\alpha}{2\beta} + 1\right)k_s + N > 0.$$

Solving the inequality for $k_s < N$ yields

$$\begin{aligned} k_s \quad < \quad & N + \frac{1}{2} + \frac{\beta}{\alpha} - \sqrt{\left(N + \frac{1}{2} + \frac{\beta}{\alpha}\right)^2 - 2N\frac{\beta}{\alpha}} \\ = \quad & \frac{2N\frac{\beta}{\alpha}}{N + \frac{1}{2} + \frac{\beta}{\alpha} + \sqrt{\left(N + \frac{1}{2} + \frac{\beta}{\alpha}\right)^2 - 2N\frac{\beta}{\alpha}}}. \end{aligned} \tag{4.38}$$

Also notice that

$$\frac{\beta}{\alpha} = \frac{T_F^{ser}/n_F}{T_G^{ser}/n_G} = \frac{T_F^M}{T_G^M} := \gamma \gg 1.$$

Thus, we obtain

$$k_s < \gamma, \text{ for } 1 \ll \gamma \ll N, \tag{4.39}$$

and

$$k_s \lessapprox \frac{1}{2}N, \text{ for } 1 \ll N \approx \gamma, \tag{4.40}$$

and

$$k_s < N, \text{ for } 1 \ll N \ll \gamma. \tag{4.41}$$

Case III: When $n_G < M < N * n_F$, we assume $n_G \leq n_F$ and consider the case that $p * n_F \leq M < (p+1) * n_F$ for some $1 \leq p < N - 1$. Then

$$
\begin{aligned}
T_{ep}^s &= \frac{(2N - k_s + 1)k_s}{2} T_G^{n_G}(\Delta T) + \left(T_F^{M/N}(\Delta T) + \cdots + T_F^{M/(p+1)}(\Delta T) \right) \\
&\quad + \left(T_F^{M/p}(\Delta T) + \cdots + T_F^{M/(N-k_s+1)}(\Delta T) \right) \\
&= \frac{(2N - k_s + 1)k_s}{2Nn_G} T_G^{ser}(T) + \left(\frac{N}{M} T_F^{ser}(\Delta T) + \cdots + \frac{p+1}{M} T_F^{ser}(\Delta T) \right) \\
&\quad + (k_s + p - N) T_F^{n_F}(\Delta T) \\
&= \frac{(2N - k_s + 1)k_s}{2Nn_G} T_G^{ser}(T) + \frac{(N + p + 1)(N - p)}{2NM} T_F^{ser}(T) + \frac{k_s + p - N}{Nn_F} T_F^{ser}(T),
\end{aligned}
$$

and

$$
\begin{aligned}
\frac{T_{ep}^s}{T_F^M(T)} &= n_F \frac{T_{ep}^s}{T_F^{ser}(T)} \\
&= \frac{(2N - k_s + 1)k_s}{2N} \frac{\alpha}{\beta} + \frac{(N + p + 1)(N - p)}{2N} \frac{n_F}{M} + \frac{k_s + p - N}{N} \\
&\leq \frac{(2N - k_s + 1)k_s}{2N} \frac{\alpha}{\beta} + \frac{(N + p + 1)(N - p)}{2N} \frac{1}{p} + \frac{k_s + p - N}{N} \\
&\overset{\text{want}}{<} 1.
\end{aligned}
$$

That is,

$$
\frac{\alpha}{2\beta} {k_s}^2 - \left(\frac{(2N + 1)\alpha}{2\beta} + 1 \right) k_s + \frac{-N^2 - N + 4Np + p - p^2}{2p} > 0.
$$

Inequality holds when

$$
k_s < N + \frac{1}{2} + \frac{\beta}{\alpha} - \sqrt{\left(N + \frac{1}{2} + \frac{\beta}{\alpha} \right)^2 - \frac{\beta}{\alpha} \cdot \frac{-N^2 - N + 4Np + p - p^2}{p}}. \tag{4.42}
$$

Notice that the right hand side of (4.42) is decreasing w.r.t $1 < p < N$. When $p = N$, it is reduced to case I when $p = 1$ and case II when $p = N$. We can also check that

when $p = N/2$, it becomes

$$k_s < N + \frac{1}{2} + \frac{\beta}{\alpha} - \sqrt{\left(N + \frac{1}{2} + \frac{\beta}{\alpha}\right)^2 - \frac{\beta}{\alpha} \cdot (\frac{3}{2}N - 1)},$$

which requires fewer iterations k_s compared to (4.38) in case II.

We summarize our discussion above and give a conclusion on the performance of EPA in the following Remark.

Remark 4.15. *The Extended Parareal Algorithm, compared to pure space paralization, has the potential to perform faster on large number of processors. More specifically, when the number of processors is limited, as in case I, one cannot expect EPA to perform faster than pure space parallelization. In this case, we suggest to use fewer time segments. On the other hand, if sufficiently many processors are available, then EPA could be faster, as in case III, provided the number of iterations is bounded by (4.42) and (4.39) - (4.41) in case II.*

Chapter 5

NUMERICAL SIMULATIONS

We implemented the numerical algorithms described in Chapter 3 to solve the electrical propagation model (3.1) with Luo-Rudy phase I (1991) model numerically in one dimension and in two dimensions.

§5.1 describes our one-dimensional experiments. We compare the performance of space-only, time-only (Standard Parareal), and space-and-time parallelization (Extended Parareal) schemes in §5.1.1-§5.1.4. Biologically-oriented simulations are discussed in §5.1.5. We study the effect of varying the stimulus, R_{gap}, and $[K]_o$. Two-dimensional simulations are reported in §5.2, where we vary R_{gap} and $[K]_o$.

The parameters $C_m = 1.2 \mu F/cm^2$ and $a = 10.0 \mu m$ were fixed in all one- and two-dimensional simulations. The rest of the parameters listed in Table 4.3 were varied for various simulation purposes.

The numerical experiments reported in this chapter were performed on the Frost cluster of the National Institute for Computational Science (NICS) at Oak Ridge National Laboratory. Frost is equipped with 2048 Intel Xeon 2.8 GHz processors (128 16-core nodes), infiniband interconnect, and gigabit ethernet network.

5.1 One-dimensional Numerical Experiments

Experiments in §5.1.1 - §5.1.3 are devote to performance of the space-only, time-only, and space-and-time parallel schemes developed in Chapter 4.

The simulations were performed on a (one-dimesional) "short" cable of length $16mm$ with parameters and initial values shown in Table 5.1. We chose a "short" cable in order to push each parallel scheme to its limit with more than enough processors and observe when the parallel schemes begin to lose in speedup and efficiency. We call this cable "short" for our parallel computing even though this cable consisted of 1000 control volumes and took more than 2 million time steps to evolve to 500 ms for non-adaptive explicit schemes, due to its fine spatial discretization of $\Delta x = 16\ \mu m$. So it is relatively "short" for parallel computing but it is costly and would take hours to accomplish with serial computing.

A single stimulus was applied on the cable at time $10\ ms$, to eliminate the influence of multiple stimuli on the action potential duration (APD).

Table 5.1: Input parameter values for 16 mm cable

Parameter	Value	Stimulus	
tmax	$500\ ms$	Period	$999999\ ms^{**}$
Δx	$16\ \mu m$	Duration	$3\ ms$
$Ri[0]$	$150\ \Omega cm^*$	Start	$10\ ms$
$Ri[1]$	$150\ \Omega cm^*$	End	$999999\ ms^{**}$
APD[0]	100	Range	$30\ \mu m$
APD[1]	900	Amplitude	$-500\ \mu A/cm^2$
Inital Vaules for Luo-Rudy (1991) Model			
V	$-84.54799678131609\ mV$	d	0.00297744387045
m	0.0016645202522	f	0.99998123976333
h	0.98330219790334	X	0.00564346929716
j	0.98952187383458	Cai	0.00017836352927

* values are set to be identical to prevent random cell lengths
** values are set large to generate a single stimulus

5.1.1 Experiments with Space Parallelization

Three fastest space parallelized solvers, **STS**, **DF** and **RK4**, were chosen to demonstrate the performance of space parallelized schemes. Recall that **STS** is the Super-Time-Stepping scheme (§3.1), here applied with $N = 4$ super-steps and damping $\nu = 0.08$; **DF** is the DuFort-Frankel scheme (§3.2); and **RK4** is the classical 4th order Runge-Kutta scheme (§3.3).

In these simulations, the three solvers were run on $16\,mm$ cable, with $\Delta x = 16\,\mu m$, yielding a mesh of 1000 control volumes, and time step $\Delta t = 0.000244\,ms$ to satify the CFL condition for these non-adaptive explicit schemes.

The CPU time, speedup and efficiency achieved by these three solvers on increasing number of processors are shown in Tables 5.2 - 5.4. The computed biological quantities are listed in Table 5.5. To compare their performance, log-scale plots of speedup and efficiency of the solvers are shown in Figure 5.1.

Table 5.2: Performance of space parallelized STS solver (with $N = 4$, $\nu = 0.08$)

Procs	CPU Time (s)	Speedup	Efficiency
1	508	—	—
2	255	1.99	99.6%
5	103	4.93	98.6%
10	85	5.98	59.8%
25	36	14.11	56.4%
50	25	20.32	40.6%
100	14	36.29	36.3%
200	10	50.8	25.4%
250	10	50.8	20.3%
500	12	42.33	8.5%
800	20	25.4	3.2%
1000	235	2.16	0.2%

We draw the following conclusions based on these experiments:

- As seen in Table 5.5, all three solvers produce identical values for the biological quantities (except small variations in dV/dt_{max}). Since **STS** is the fastest among

Table 5.3: Performance of space parallelized DF solver

Procs	CPU Time (s)	Speedup	Efficiency
1	1132	—	—
2	563	2.01	100.5%
5	255	4.44	88.8%
10	191	5.93	59.3%
25	81	13.98	55.9%
50	51	22.20	44.4%
100	29	39.03	39%
200	20	56.6	28.3%
250	22	51.45	20.6%
500	23	49.22	9.8%
800	31	36.52	4.6%
1000	34	33.29	3.3%

Table 5.4: Performance of space parallelized RK4 solver

Procs	CPU Time (s)	Speedup	Efficiency
1	10306	—	—
2	5153	2	100%
5	2221	4.64	92.8%
10	1656	6.22	62.2%
25	735	14.02	56.1%
50	433	23.80	47.6%
100	275	37.48	37.5%
200	189	54.53	27.3%
250	203	50.77	20.3%
500	244	42.24	8.4%
800	448	23	2.9%
1000	378	27.26	2.7%

Table 5.5: Computed biological quantities from STS, DF and RK4

Solver	APD0 (ms)	APD1 (ms)	Spd (cm/s)	Vmax (mV)	dV/dt_{max} (mV/ms)
STS	386.00	380.00	98.462	38	437
DF	386.00	380.00	98.462	38	434
RK4	386.00	380.00	98.462	38	436
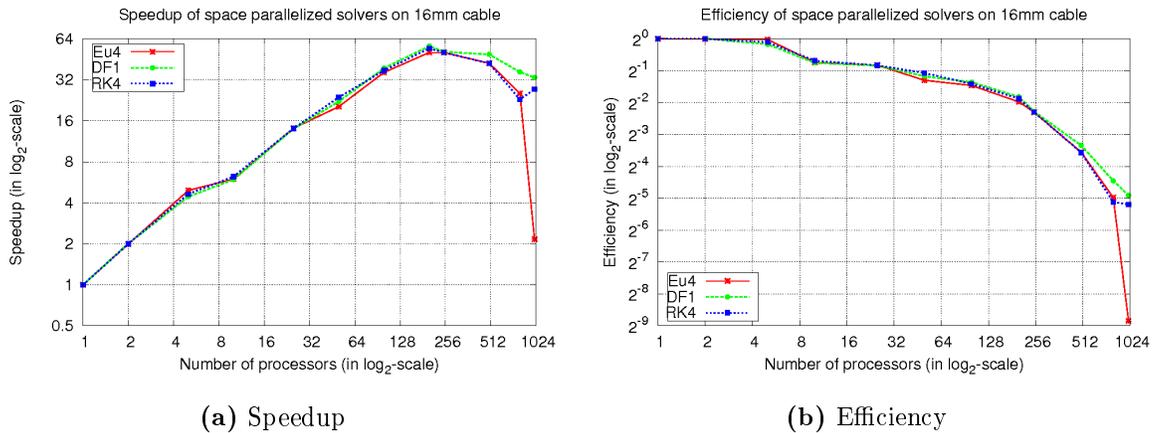
Figure 5.1: Speedup and efficency of space parallelized STS (red),
DF (green) and RK4 (blue), in log-scales

these three (almost 2 times faster than **DF** and 20 times faster than **RK4**), **STS** is the best candidate for coarse propagator in the Parareal Algorithm.

- Speedup and efficiency achieved by space parallelization of these three solvers remain essentially linearly increasing for up to about 200 processors, as expected by Amdahl's Law on fixed size problem. After that, both speedup and efficiency decrease quickly.

- All these three solvers achieve their best speedup on 200 processors and then begin to lose. Therefore, to achieve the fastest computation on this 16 mm cable using these three solvers, the optimal number of processors to be used for space parallelization is 200. This number will be used in later experiments of testing the parareal algorithms.

- The fact that these three solvers achieve their best performance (best CPU time) on the same number of processors and their speedup and efficiency curves stay close to each other is unexpected. In fact, this is caused by the high-speed communication network of the Frost cluster. Similar tests on a (small) cluster with slower interconnect produced curves noticeably away from each other. Thus, communication hardware can affect performance significantly.

These experiments on a fixed-size problem clearly show the trouble with space parallelization. It performs well and achieves linear (or nearly linear) speedup and best (or nearly best) efficiency only on up to a certain number of processors, as expected from Amdahl's Law. With more processors, performance dropped off. When extremely many processors were used (> 500 processors here), in which case the cost of communication between processors contributed significantly in the overhead, both speedup and efficiency dropped dramatically.

5.1.2 Experiments with Time Parallelization

Another series of experiments were performed with the Standard Parareal Algorithm (SPA) for time-only parallelization on the same 16 mm cable. The combination of serial **STS** (with $N = 4$, $\nu = 0.08$) and serial **RK45** were selected for coarse and fine propagators, respectively. A tolerance, $TOL = 2.0$, was set to be the infinite norm of the difference of computed values, including both voltage and gates, obtained at the end of all time segments in two successive iterations and used to terminate parareal iteration. The results are shown in Table 5.6. The computed biological quantities are almost identical to those listed in Table 5.5.

Table 5.6: Performance of Standard Parareal with $TOL = 2$: serial STS and RK45 combination

Coarse	Fine	Procs (= time segments)	CPU (s)	Speedup	Efficiency
RK45	—	1	2157	—	—
STS	RK45	1	2665	0.81	80.9%
STS	RK45	2	1873	1.15	57.6%
STS	RK45	5	1370	1.57	31.5%
STS	RK45	10	1401	1.54	15.4%
STS	RK45	50	1285	1.68	3.4%
STS	RK45	100	1901	1.13	1.1%
STS	RK45	125	1882	1.15	0.9%
STS	RK45	250	1883	1.15	0.5%
STS	RK45	500	1863	1.16	0.2%

The observed performance of the Standard Parareal Algorithm in solving our problem is not as promising as we expected. It suffers both poor performance and poor scaling. In fact, compared to space-only parallelization (Tables 5.2, 5.4), the CPU time and speedup are much worse, on small and large number of processors alike.

We investigated and analyzed the computational cost of the coarse and fine propagators separately and found that the high overall computational cost is due to the coarse propagator, not the fine propagator, despite the fact that the coarse propagator is cheap and fast relative to the fine propagator. In one iteration of the SPA algorithm, the coarse propagator runs sequentially, one time slice after another, from $tstart$ to $tmax$, on the entire cable, on one processor. Thus, the cost of the serial coarse propagator remains the same essentially in every iteration no matter how many processors or time-segments are allocated. On the contrary, more time segments reduce the cost of the fine propagator on one iteration since they run concurrently. Therefore, the more time segments the total time is divided into, the higher the contribution of the coarse propagator in the cost.

The observation and analysis of our experiments confirm comments made by other researchers in the literatures [50] about the Standard Parareal Algorithm, namely that, although it is applicable theoretically, it is not recommended to use a larger number of processors/segments.

On the other hand, it is our high hope that the Extended Parareal we proposed can overcome this drawback of the Standard Parareal and achieve much better performance on a large number of processors, by replacing the serial propagators with their space parallelized ones to accelerate both coarse and fine solvers. This leads us to another series of experiments, described in the next subsection.

5.1.3 Experiments with Time-and-Space Parallelization

To compare the performance of the Extended Parareal Algorithm (EPA) with its standard cousin, numerical experiments with the same combination of algorithms **STS - RK45**, were conducted on the same 16 mm cable again. This time, the coarse solver **STS** was space-parallelized and **RK45** remained serial in space (since, being adaptive, it cannot be paralllelized in space). An optimal number of processors = 200 was set for space parallelization of the coarse solver STS in the code (see Table 5.2). That is, all processors will be used if the total number of processors is less than 200; otherwise, 200 processors will be used for the space parallelized **STS**.

The results are listed in Table 5.7 and we compare them with our experiments of Table 5.6 in Figure 5.2. It is clear that EPA performs *much better* than SPA by any measure (CPU time, speedup, efficiency, scaling), as anticipated in Remark 4.15. EPA maintains nearly linear speedup up to 500 processors, whereas SPA flattens out beyond just 5 processors.

Table 5.7: Performance of Extended Parareal with $TOL = 2$: space-parallelized STS and RK45 combination

Coarse	Fine	Procs (= time segments)	CPU (s)	Speedup	Efficiency
STS	RK45	1	2674	0.81	80.7%
STS	RK45	2	1606	1.34	67.2%
STS	RK45	5	689	3.13	62.6%
STS	RK45	10	529	4.08	40.8%
STS	RK45	25	217	9.94	39.8%
STS	RK45	50	119	18.13	36.3%
STS	RK45	100	119	18.13	18.2%
STS	RK45	125	99	21.79	17.4%
STS	RK45	250	69	31.26	12.5%
STS	RK45	500	64	33.7	6.7%

Furthermore, in order to compare the performance of a space-parallelized numerical scheme working as the fine propagator in the EPA with its purely space-parallelized version, we selected the space-parallelized **RK4** as a good candidate.

(a) Speedup **(b)** Efficiency
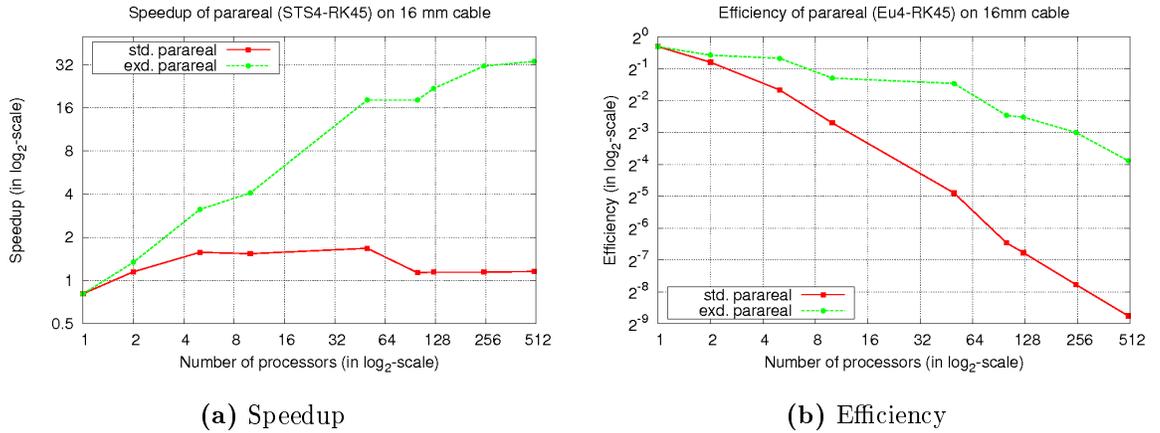
Figure 5.2: Comparison of the Standard (red) and Extended (green)
Parareal Algorithms

Thus, the EPA used the combination **STS - RK4**. Both propagators were run with the optimal number of processors, 200, based on our previous experiments (see Table 5.2 and Table 5.4). The results are shown in Table 5.8 and Figure 5.3.

Table 5.8: Performance of the Extended Parareal with $TOL = 2$:
space-parallelized STS and RK4 combination

Coarse	Fine	Procs	Time Segments	CPU (s)	Speedup	Efficiency
STS	RK4	50	1	559	18.44	36.9%
STS	RK4	200	1	239	43.12	21.6%
STS	RK4	400	2	158	65.23	16.3%
STS	RK4	800	4	115	89.62	11.2%
STS	RK4	1000	5	101	102.04	10.2%

Comparing the Extended Parareal and space-parallelized cousins, the performance of the Extended Parareal is outstanding, especially when a large number of processors is at our disposal, and greatly saves computational time. This merit makes the Extended Parareal a more suitable approach for future large scale simulations.
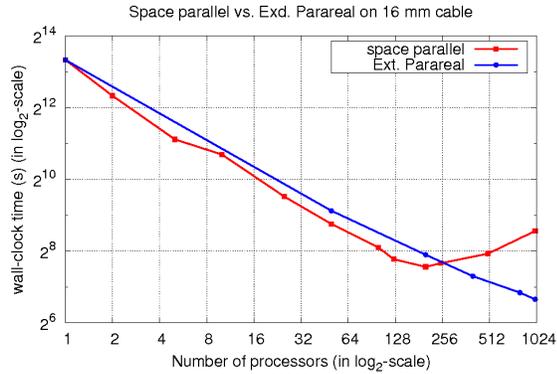
Figure 5.3: Comparison of space-only parallelized RK4 (red) and the Extended Parareal Algorithm, STS-RK4 combination (blue)

5.1.4 Summary of Comparisons of Parallel Schemes

Summarizing our results of the previous sections of this chapter and comparing the three parallel schemes, we point out some interesting observations and draw some important conclusions on the use of the numerical and parallel schemes in this section.

- As indicated in Table 5.5 and other tests we performed, all numerical solvers produce identical (or nearly identical) values for those biological quantities.

- Since **STS** is the fastest solver, it is the best candidate for coarse propagator in the Standard and Extended Parareal Algorithms.

- On small number of processors, the performance of the Extended Parareal is comparable to that of the space-parallelized schemes, and far better than that of the Standard Parareal; When larger number of processors are available, the Extended Parareal Algorithm performs much better than the other two, which makes it the best parallel scheme among these three schemes. Our numerical experiments match the performance discussions described in Remark 4.15.

- The experiments on the EPA using the **STS - RK45** combination, described in §5.1.3, show that the Extended Parareal can be viewed as a vehicle to

84

employ adaptive schemes (which are impossible to parallelize in space due to synchronization difficulties) in parallel computing.

The excellent performance of the Extended Parareal Algorith made it possible to conduct extensive biologically oriented simulations.

5.1.5 Biologically-oriented Simulations

Extensive biological simulations and parametric studies have been performed on cables of various lengths using the Extended Parareal Algorithm, in which we systematically vary some of the main parameters in the model.

In this section we describe simulations on a cable of length 50 mm. This length is a good compromize between problem size and computational cost. It is long enough to be biologically realistic, too long for serial computing (several hours each, see Chapter 3), but manageable with parallel computing (on hundreds of processors!). In our parametric studies we varied the following parameters:

1. External concentration of potassium $[K]_o$. The normal value is $[K]_o = 5.4 \, mM$. We tested both lower values ($=3, 4$) simulating conditions of **hypokalemia**, and higher values ($=7, 8, 9, 10, 11, 12, 13$) simulating conditions of **hyperkalemia**.

2. Gap junction resistivity, R_{gap}, which is applied at boundaries between cardiac cells. The normal cytoplasmic resistivity for human myocytes is $Ri = 150 \, \Omega cm$ and R_{gap} is considerably higher, but no precise values are known. We tested several higher values: $R_{gap} = 300, 600, 1000, 2000 \, \Omega cm$.

3. Single or repeated (every $200 \, ms$) stimulus was applied in a small ($10 \, \mu m$) region at the left end of the cable.

Their influence on the biological quantities, such as action potential duration (APD), propagation speed (*speed*) and maximal potential (V_{max}) were monitored and will be presented below. To visualize the propagation of action potentials, we

85

output V at five equispaced points along the cable (at $x = 0, L/4, L/2, 3L/4, L$, with $L = $ cable length).

Each value of $[K]_o$, assumed to hold everywhere, determines a resting (steady) state which we compute by a simulation, starting from arbitrary initial values, *without applying any stimulus*. The computed rest values for the potential V and the gates are listed in Table 5.9. They are used as initial values for subsequent simulations.

The first simulation, on a cable of length $50\,mm$, is with (normal) $[K]_o = 5.4\,mM$, and $Ri = R_{gap} = 150\,\Omega cm$. Voltage history at five equally spaced nodes is shown in Figure 5.4a. Note that the curves are not sigular at the upstrokes. A zoom-in plot at the upstroke of the action potential at the right-end of the cable is shown in Figure 5.4b. Voltage profiles at the first node is shown in Figure 5.5a and corresponding gates history in Figure 5.5b.
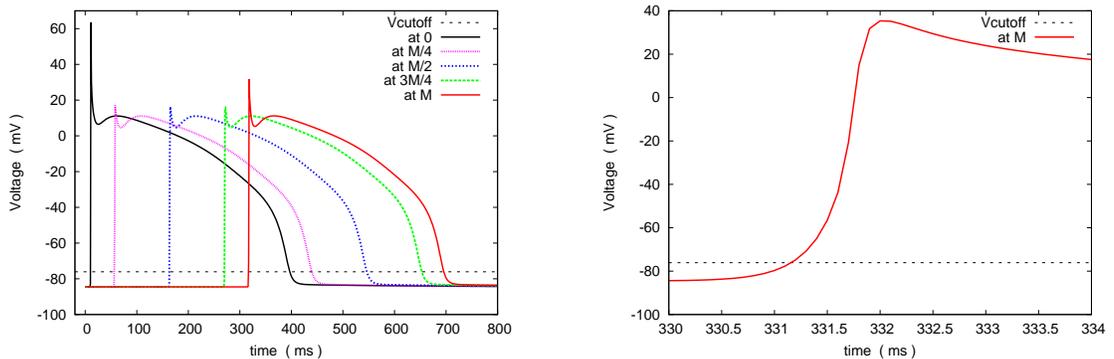
(a) Voltage history at 5 equally spaced nodes   (b) Zoom-in at upstroke of AP at right-end of cable

Figure 5.4: Action potential on a cable of length $50\,mm$

In the first set of tests, we apply stimulus of various strengths on the same cable. The obtained results perfectly matched the description of "all-or-nothing" property and produced either identical action potentials (Figure 5.4a), when the strength was above certain threshold, or flat lines when the strength was below the threshold.

The next series of tests were designed to manipulate the stimulus period and observe its effect on the potential propagation. Some results are shown in Figure 5.6.

86

Table 5.9: $[K]_o$ and associated potential and gates in Luo-Rudy phase I (1991) model

| $[K]_o$	5.4 (normal)	3.0 (abnormal)	4.0 (abnormal)
V	-84.54799678282664	-96.00223575957419	-90.81551303451198
m	0.00166648217313	0.00024070076809	0.00058200973949
h	0.98330219789904	0.99872727894491	0.99590013700613
j	0.98952187383367	0.99889732941648	0.99703695298703
d	0.00297744387078	0.00114649813087	0.00176092718051
f	0.99998123976333	0.99999782305501	0.99999422716513
X	0.00564346895260	0.00180173175262	0.00303206018008
Cai	0.00017836352928	0.00013243547761	0.00014830982920
$[K]_o$	7.0 (abnormal)	8.0 (abnormal)	9.0 (abnormal)
V	-78.67973684400134	-75.54165652692299	-72.70773238255576
m	0.00438066429310	0.00728002211552	0.01144339635209
h	0.93986515149662	0.88490560164669	0.80201421451890
j	0.95809689756957	0.90723546645185	0.82153559019045
d	0.00490745454890	0.00643453367652	0.00823835750666
f	0.99994345209289	0.99989797475749	0.99982613252835
X	0.00998842432493	0.01348050680379	0.01760223313960
Cai	0.00022359447247	0.00025791194364	0.00029720770210
$[K]_o$	10.0 (abnormal)	11.0 (abnormal)	12.0 (abnormal)
V	-70.12397264679385	-67.74748787738635	-65.54718840722101
m	0.01717608198246	0.02479289747129	0.03460798138824
h	0.69306674613317	0.56882167821468	0.44498255576699
j	0.70569924515929	0.57458900871432	0.44585024710676
d	0.01034236960485	0.01277070476184	0.01554927719980
f	0.99971727689208	0.99955792433128	0.99933118604069
X	0.02236497194614	0.02777227685655	0.03382343582322
Cai	0.00034169540110	0.00039158233702	0.00044709573228
$[K]_o$	13.0 (abnormal)		
V	-63.50059136465195		
m	0.04690119914541		
h	0.33513708858460		
j	0.33317233223800		
d	0.01870109003990		
f	0.99901664361355		
X	0.04050452208904		
Cai	0.00050838365402		
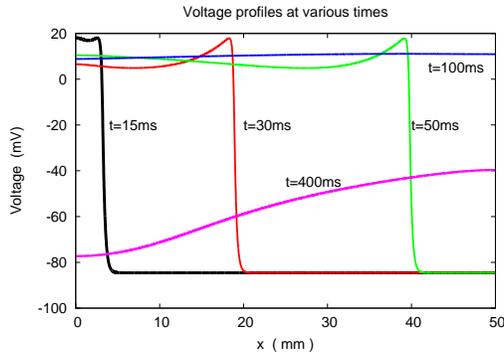
(a) Voltage profiles at 15, 30, 50, 400 ms
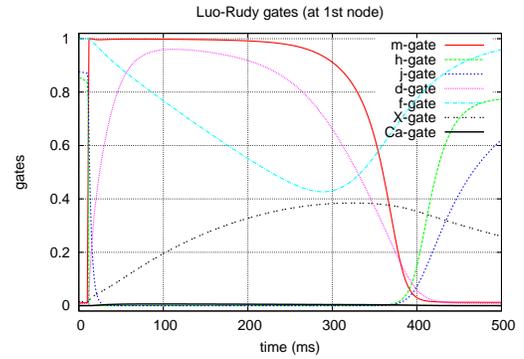
(b) Gates vs time.

Figure 5.5: Voltage profiles and gates history at the first node on a cable of length 50 mm
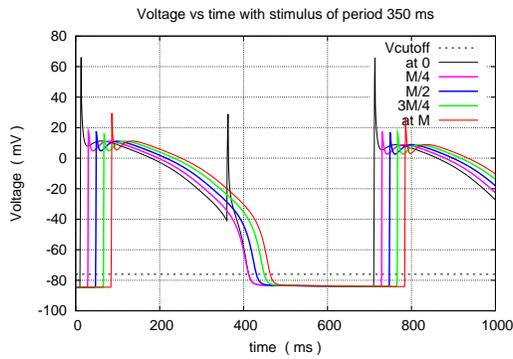
(a) Frequent stimulations of period 350 ms
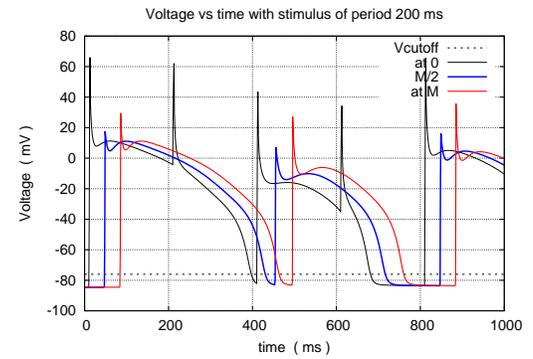

(b) Frequent stimulations of period 200 ms

**Figure 5.6:** Simulation of refractoriness on a cable of length 50 mm

The action potential duration shown in Figures 5.6a - 5.6b is about $400\,ms$. From Figure 5.6a, one can see that the 2nd action potential is evoked by the 3rd stimulus, which takes place after the absolute refractory period (ARP) of the 1st stimulus, rather than the 2nd stimulus occurring inside the APR of the 1st stimulus. The same phenomenon can be observed in Figure 5.6b as well with stimulus period being $200\,ms$ so that the 2nd stimulus takes place right after the ARP. Figure 5.6 demonstrates refractoriness of the action potential, as well as the robustness of its shape.

Next, we study the effect of R_{gap} on the main biological quantites. Results at the middle of the cable appear in Figures 5.7a - 5.7d. From these figures, we can see that (a) R_{gap} does not change the shape of the actions potential, but higher R_{gap} delays the upstroke, as expected. (b) The action potential duration is very robust and R_{gap} has little effect on it. APD remains almost constant as R_{gap} is varied. (c) Higher R_{gap} decreases the propagation speed. A decaying cubic (in green): $y = x^{-3}$ fits it very well. (d) R_{gap} increases V_{max} significantly. A cubic (in green): $y = x^3$ gives a good fit.

The above simulations were with normal $[K]_o = 5.4$. To detect the effect of R_{gap} and of abnormal $[K]_o$, we repeated the above simulations with various $[K]_o$ and R_{gap} values. Selected representative results are shown in Figures 5.8a - 5.8d. They show that R_{gap} has similar effect as in the normal $[K]_o$ case, and that abnormal $[K]_o$ affects APD, propagation speed and V_{max} significantly.

Finally, we explore what happens if only a portion of the cable is exposed to abnormal $[K]_o$. For each of the abnormal $[K]_o$ values: 3, 4, 7, 8, 9, 10, 11, 12, 13 mM, we performed seven simulations, with the abnormal value applied to a different portion of the cable and in different parts, e.g. in the left third, the right third, left two thirds, middle two thirds, right two thirds, etc. This set of 63 simulations attempts to capture the variability of APD, propagation speed and V_{max}, which are visualized by the red vertical segments in Figures 5.9a - 5.9c.

89
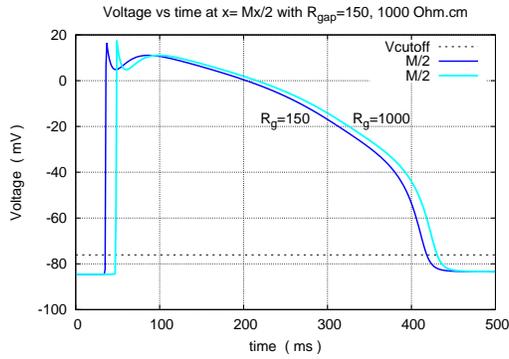
(a) Effect of R_{gap} on potential
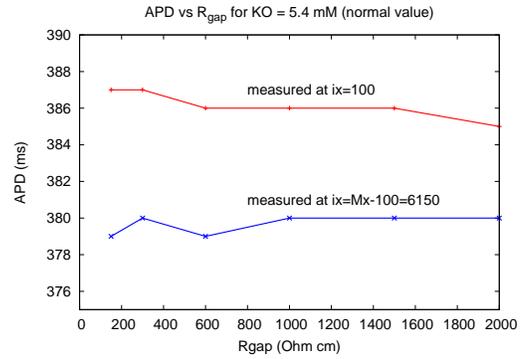
(b) Effect of R_{gap} on APD
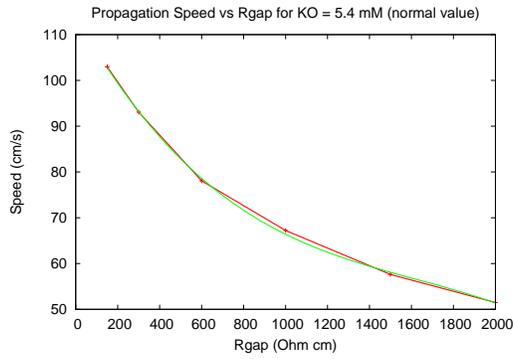
(c) Effect of R_{gap} on propagation speed
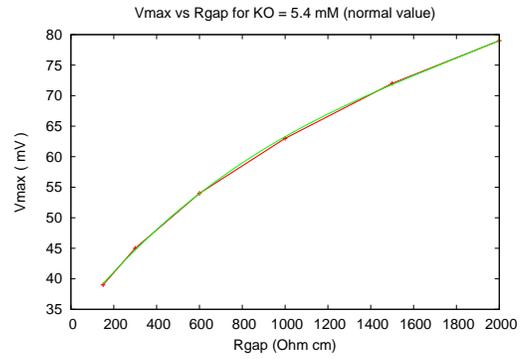
(d) Effect of R_{gap} on V_{max}

Figure 5.7: Effect of R_{gap} on biological quantities
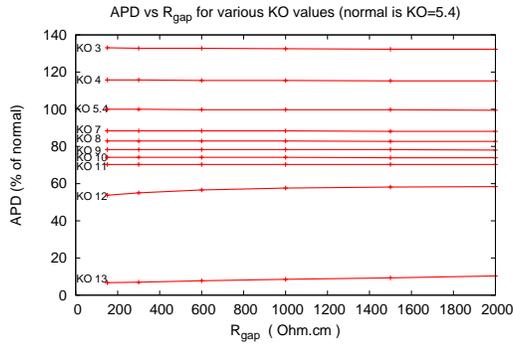
(a) Effect on APD
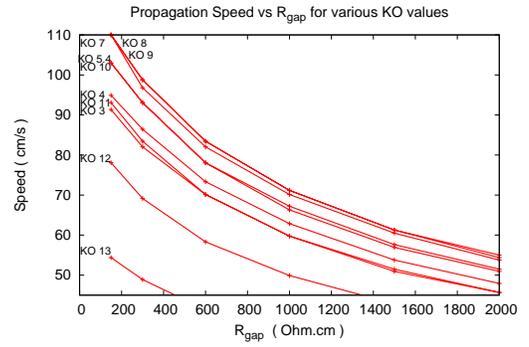
(b) Effect on propagation speed
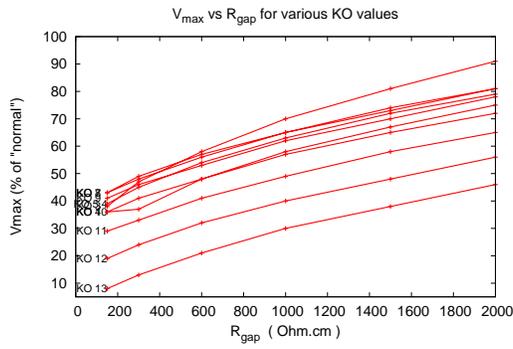
(c) Effect on V_{max}
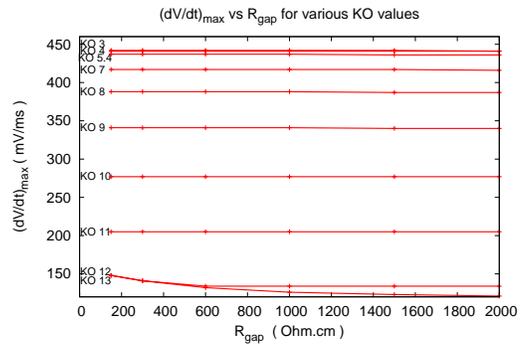
(d) Effect on dV/dt_{max}

Figure 5.8: Effect of varying R_{gap} and $[K]_o$.
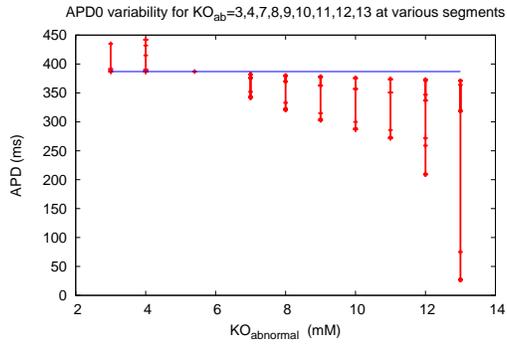
(a) Variability of APD.
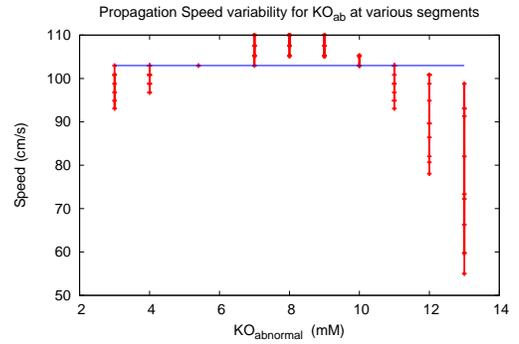
(b) Variability of propagation speed.
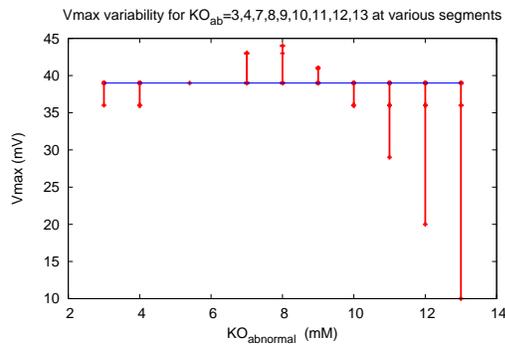
(c) Variability of V_{max}.

Figure 5.9: Effect of $[K]_o$ on biological quantities.

We see that the effect of abnormal $[K]_o$ values on APD (Figure 5.9a) is quite different from the effect on propagation speed (Figure 5.9b) and V_{max} (Figure 5.9c), which are very similar.

These experiments on one-dimensional cables led us to hope that abnormal $[K]_o$ values, placed in a portion of two-dimensional tissues, may have significant effect on the potential propagation and drive it into irregular patterns. Our two-dimensional experiments, described in the next section, verify this guess.

5.2 Two-dimensional Experiments

Our two-dimensional experiments focus on simulating abnormal conditions that may shed light on generation of arrhythmias. Arrhythmic behavior is unlikely to be captured in one space dimension, since it is often associated with chaotic and complicated patterns.

Biological experiments have shown that sick tissue is often associated with regions of abnormal potassium concentration. The condition of lower K^+ concentration in the blood is called **hypokalemia**, and that of higher concentration **hyperkalemia**. The normal value of extracellular potassium in the Luo-Rudy ionic model is $[K]_o = 5.4 \, mM$, so we consider a value of 4 as representing hypokalemia and a value of 11 for hyperkalemia.

The quantity $[K]_o$ enters several formulas in the Luo-Rudy ionic model, so in particular alters the resting state (voltage and gates). $[K]_o$ values and associated resting voltage and gates in Luo-Rudy phase I (1991) model were listed in Table 5.9.

In order to simulate hypokalemia and hyperkalemia, we assign an abnormal value $[K]_o$ (= 4 or 11) in a rectangular region of control volumes while the rest of the tissue has normal $[K]_o = 5.4 \, mM$.

The two-dimensional simulations reported below were performed on tissue of dimensions $51mm \times 31mm$, more precisely: $51200 \, \mu m \times 30720 \, \mu m$, using $\Delta x = 32 \, \mu m$ and $\Delta y = 16 \, \mu m$, resulting in a grid of 1600×1920 nodes.

For easy comparison and reference, we list all computed biological quantities in our two-dimensional experiments, column by column, in Table 5.10.

Table 5.10: Computed biological quantities in tissue of dimensions $51 \times 31mm$ with various pairs of $[K]_o$, R_{gap} values

$[\mathbf{K}]_\mathbf{o}(\mathbf{mM})$, $\mathbf{R_{gap}}(\mathbf{\Omega cm})$	5.4 , 150	5.4 , 1000	4.0 , 150	4.0 , 1000	11.0 , 1000
APD0 (ms)	383.31	382.86	196.58	382.86	282.60
APD1 (ms)	379.91	381.89	272.24	381.99	289.10
speed (cm/s)	125.5	48.8	36.8	47.4	45.3
Vmax (mV)	66	174	118	174	174
dV/dt$_\mathbf{max}$ (mV/ms)	1000	973	1000	973	998

5.2.1 Simulations of healthy tissue

Before proceeding to simulate more complicated cases, we first lay down a benchmark by examining healthy tissue of dimension $51mm \times 31mm$ without abnormal region. We simulated two cases, one with $R_{gap} = 150\,\Omega cm$ and one with $R_{gap} = 1000\,\Omega cm$. Stimulus of identical strength and period $200\,ms$ were placed at the lower-left corner.

Voltage profiles with $R_{gap} = 1000\,\Omega cm$ at several time points are shown in Figure 5.10.

Both cases produced similar action potentials, but the higher resistivity reduces the propagation speed, as expected. This can be seen in Figures 5.11a - 5.11b, showing voltage histories at three locations (one at lower-left corner inside the stimulated region, one at the center and one on the right).

The computed biological quantities are listed in columns 2 and 3 of Table 5.10. As in the one-dimensional simulations, only propagation speed and V_{max} are significantly affected by R_{gap} while the other quantities stay nearly identical.

5.2.2 Simulations of hypokalemia

The next two experiments were conducted again on tissue of dimensions $51mm \times 31mm$, with $R_{gap} = 1000\,\Omega cm$, and stimulus of period $200\,ms$ applied near the lower
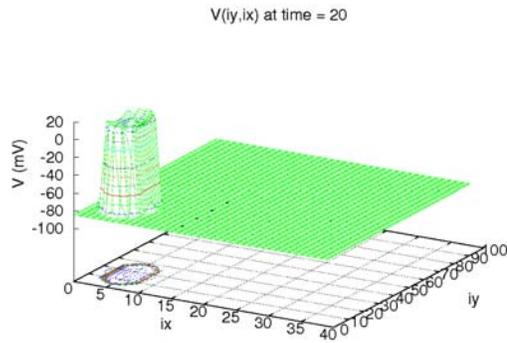
(a) Voltage at $t = 20\,ms$
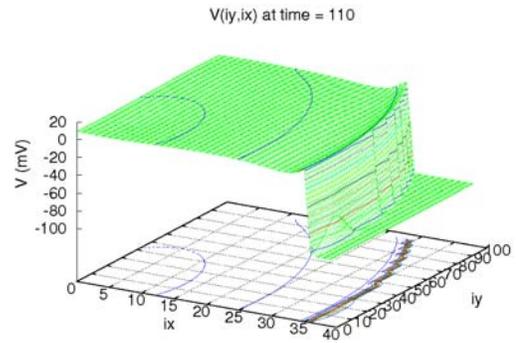
(b) Voltage at $t = 110\,ms$
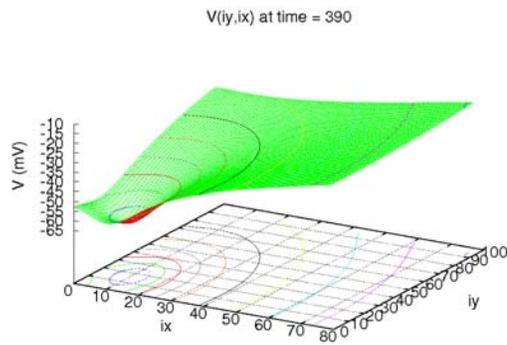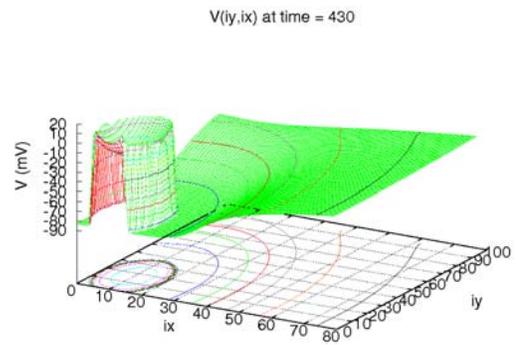
(c) Voltage at $t = 390\,ms$
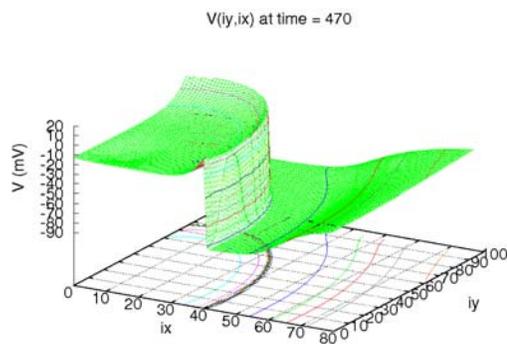
(d) Voltage at $t = 430\,ms$

(e) Voltage at $t = 470\,ms$
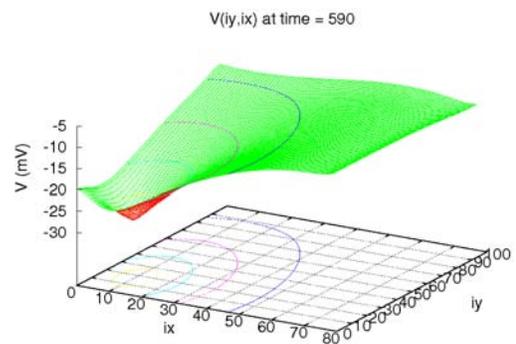
(f) Voltage at $t = 590\,ms$

Figure 5.10: Action potential propagation in tissue of dimensions $51mm \times 31mm$ with normal $[K]_0 = 5.4\,mM$ and $R_{gap} = 1000\,\Omega cm$.

95
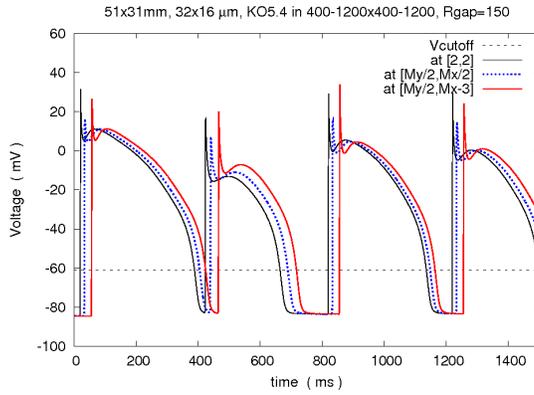
(a) $[K]_0 = 5.4\,mM$ & $R_{gap} = 150\,\Omega cm$
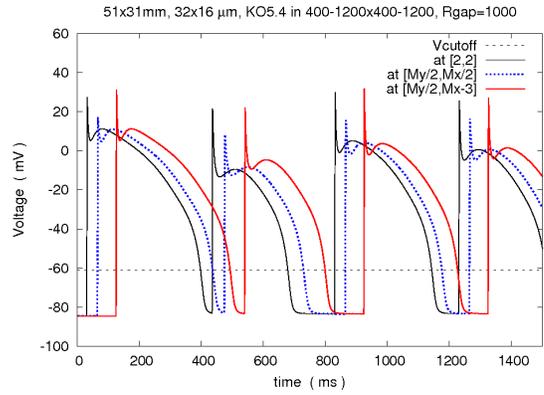
(b) $[K]_0 = 5.4\,mM$ & $R_{gap} = 1000\,\Omega cm$
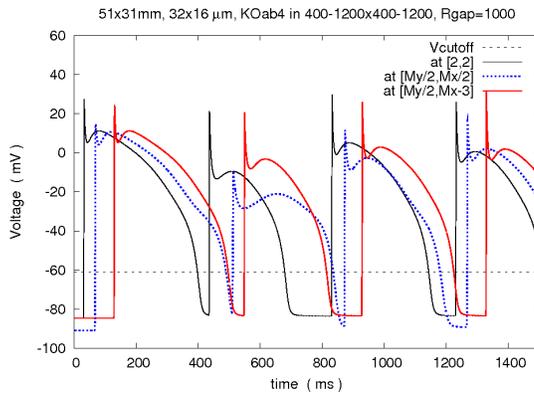
(c) $[K]_0 = 4\,mM$ & $R_{gap} = 1000\,\Omega cm$
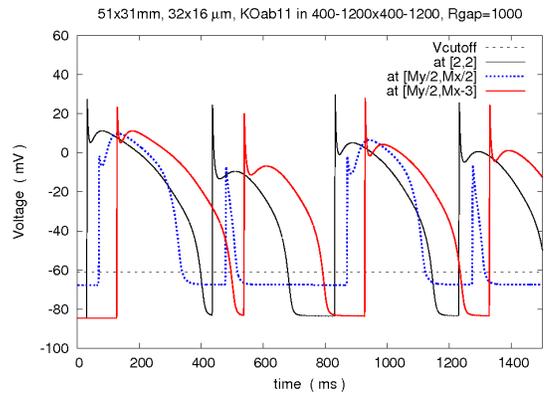
(d) $[K]_0 = 11\,mM$ & $R_{gap} = 1000\,\Omega cm$

Figure 5.11: Votage history at three identical nodes in tissue of dimensions $51mm \times 31mm$ with various $[K]_o$ and R_{gap}.

left corner. An abnormal value $[K]_o = 4.0 \, mM$ is set in a rectangle 800×800 nodes ($= 25.6 \times 12.8 \, mm$), placed off center (visible in the surface plots).

In order to clearly see that voltage propagated in a rather chaotic pattern, several snapshots are shown in Figures 5.12, 5.13, 5.14, and 5.15, after the 1st, 3rd, 4th and 6th stimulus, respectively. Several differences can be observed easily:

- In Fig.5.12a, the "dip" represents the resting values corresponding to abnormal $[K]_0 = 4.0$. It is noticeably different from Fig.5.10a of the previous experiment.

- The voltage profiles evolve into complicated, non-repeating and rather chaotic pattern after each stimulus. We believe this is a good indicator of arrhythmic behavior.

- One interesting phenomenon is that the voltage circled the abnormal region either from both sides (around time 490 ms, Fig.5.13c) and jointed together (around time 520 ms, Fig.5.13d), or from one side (around time 640 ms, Fig.5.14d).

Voltage history at three nodes, same as the ones selected in $[K]_0 = 5.4$ case, is shown in Figure 5.11c. One can see that the blue curve (voltage at the node inside the abnormal region) has quite different shape from the one in Figure 5.11b.

The computed biological quantities are shown in columns 4 and 5 of Table 5.10. Unlike the normal cases, all computed biological quantities, affected by the combination of R_{gap} and $[K]_o$, are quite different in columns 4 and 5.

5.2.3 Simulations of hyperkalemia

Simulations of hyperkalemia were performed on tissue of the same size ($51mm \times 31mm$), with the same settings as in §5.2.2, but with high abnormal $[K]_o = 11.0 \, mM$ in exactly the same rectangular region. It can be seen in Figure 5.16a that the voltage starts with a bump instead of a dip (Figure 5.12a), and propagated in quite different manner. It propagated over, not around, the abnormal region, as can be seen in Figure 5.16b and Figure 5.16f. However, the pattern is still rather chaotic. Voltage
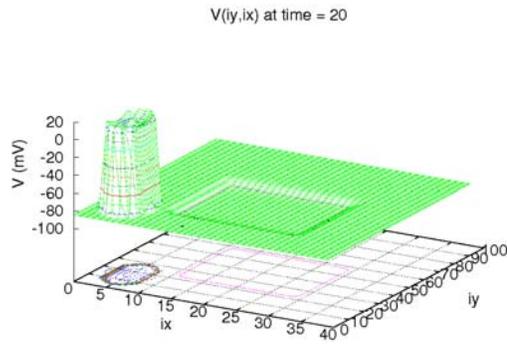
(a) Voltage at $t = 20\,ms$
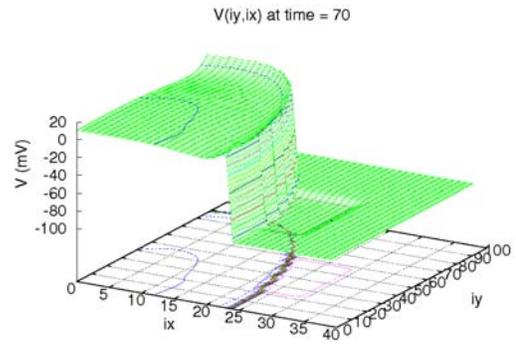
(b) Voltage at $t = 70\,ms$
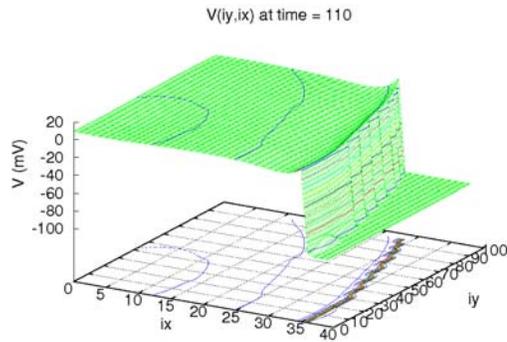
(c) Voltage at $t = 110\,ms$
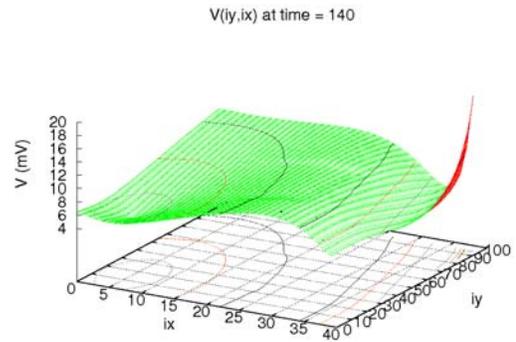
(d) Voltage at $t = 140\,ms$
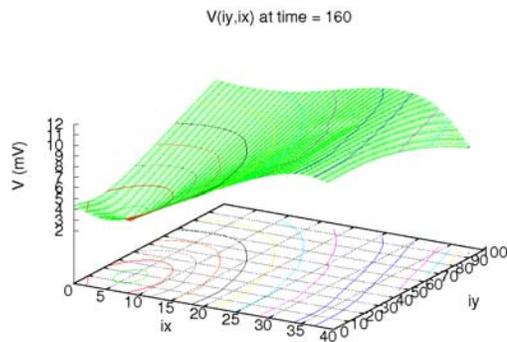
(e) Voltage at $t = 160\,ms$
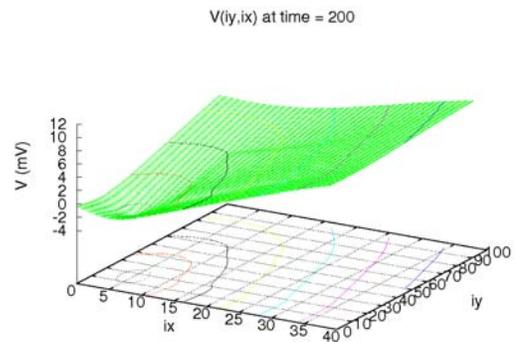
(f) Voltage at $t = 200\,ms$

Figure 5.12: Action potential propagation after the 1st stimulus in tissue of dimensions $51mm \times 31mm$ with abnormal $[K]_0 = 4.0\,mM$ in a rectangular region placed off center.
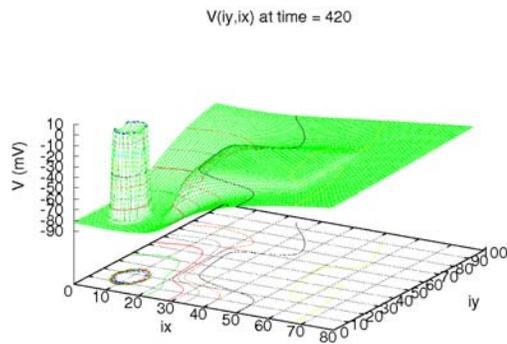
(a) Voltage at $t = 420\,ms$
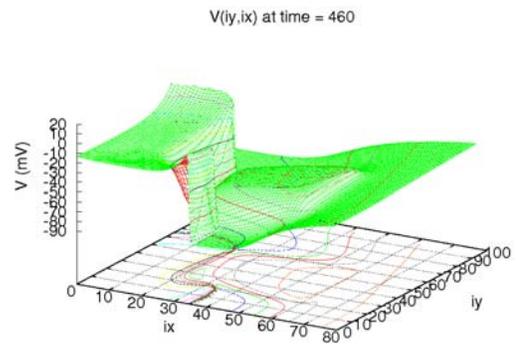
(b) Voltage at $t = 460\,ms$
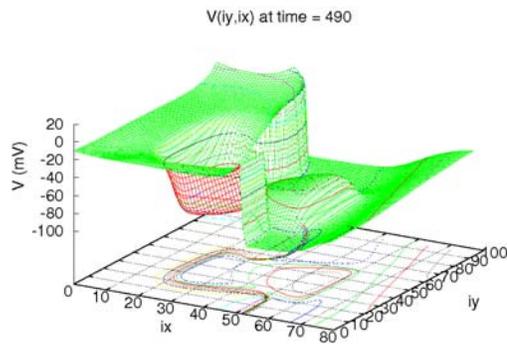
(c) Voltage at $t = 490\,ms$
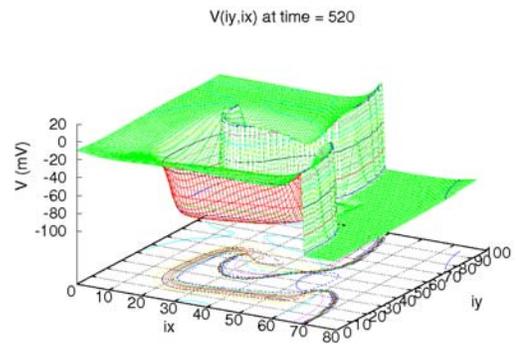
(d) Voltage at $t = 520\,ms$
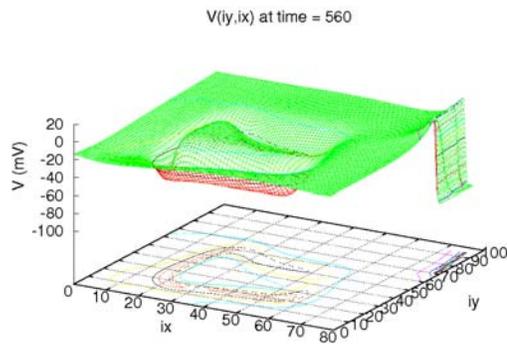
(e) Voltage at $t = 560\,ms$
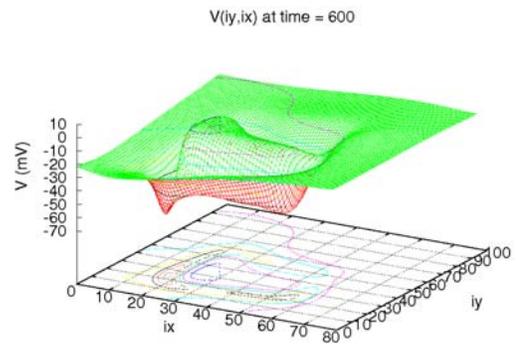
(f) Voltage at $t = 600\,ms$

Figure 5.13: Same as Figure 5.12 but after the 3rd stimulus.
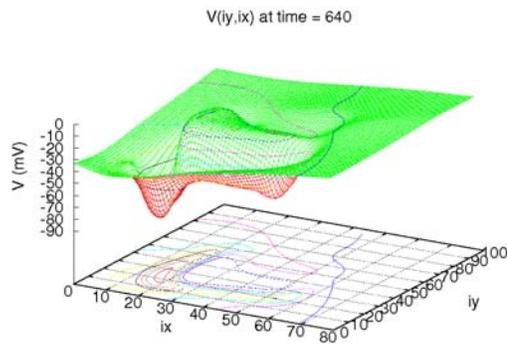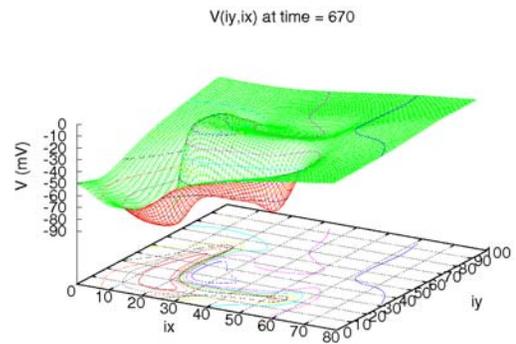
(a) Voltage at $t = 640\,ms$ (b) Voltage at $t = 670\,ms$
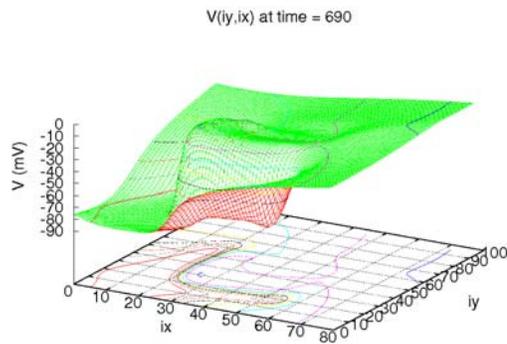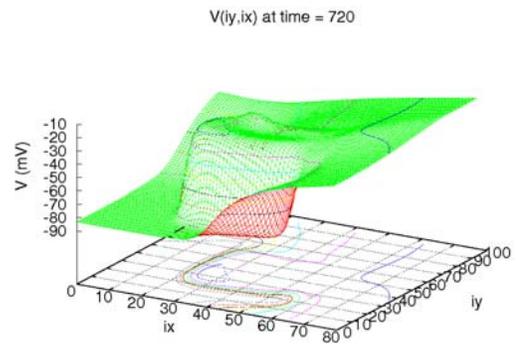
(c) Voltage at $t = 690\,ms$ (d) Voltage at $t = 720\,ms$
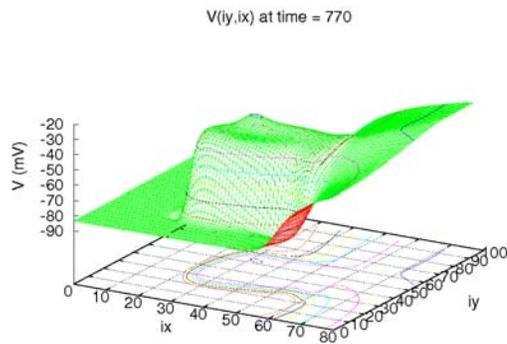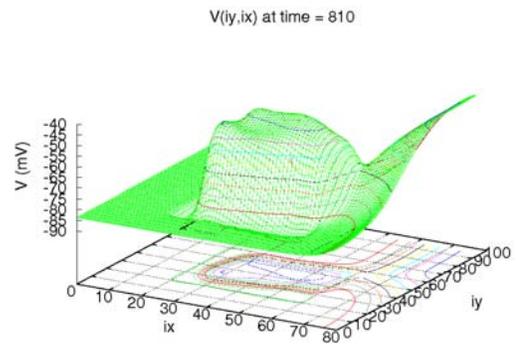
(e) Voltage at $t = 770\,ms$ (f) Voltage at $t = 810\,ms$

Figure 5.14: Same as Figure 5.12 but after the 4th stimulus.
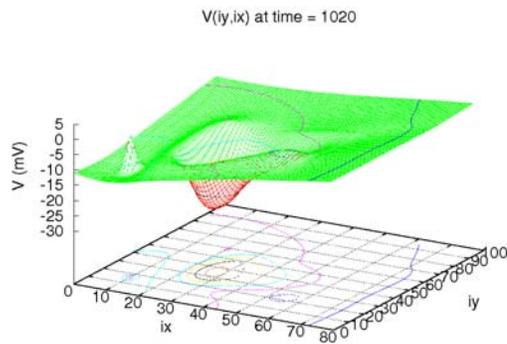
(a) Voltage at $t = 1020 \, ms$
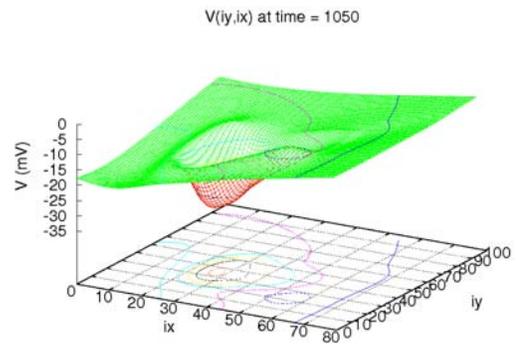
(b) Voltage at $t = 1050 \, ms$
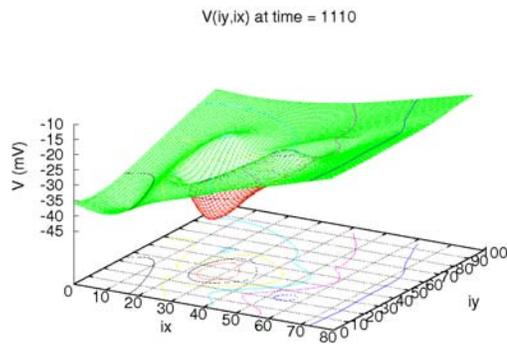
(c) Voltage at $t = 1110 \, ms$
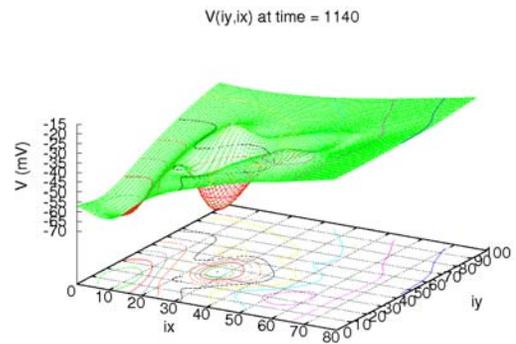
(d) Voltage at $t = 1140 \, ms$
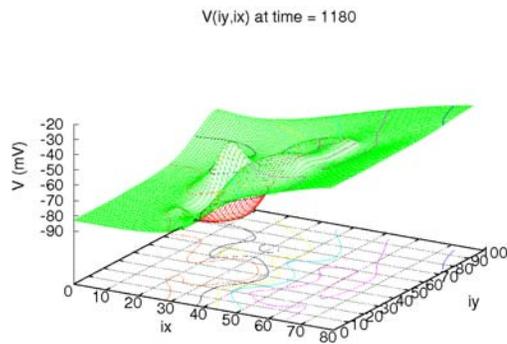
(e) Voltage at $t = 1160 \, ms$
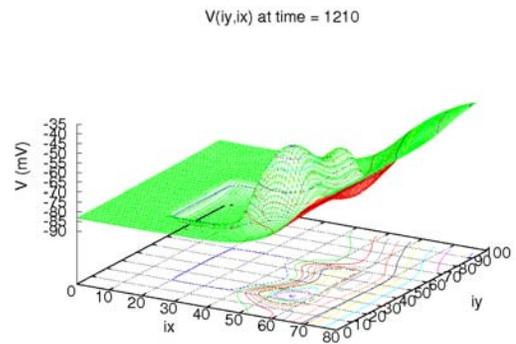
(f) Voltage at $t = 1210 \, ms$

Figure 5.15: Same as Figure 5.12 but after the 6th stimulus.

history is shown in Figure 5.11d and the computed biological quantities are shown in column 6 of Table 5.10. Again, obvious difference can be observed on blue curve due to the effect of abnormal $[K]_o$ value.
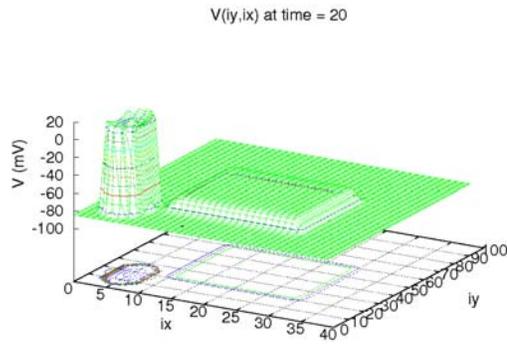
(a) Voltage at $t = 20\,ms$
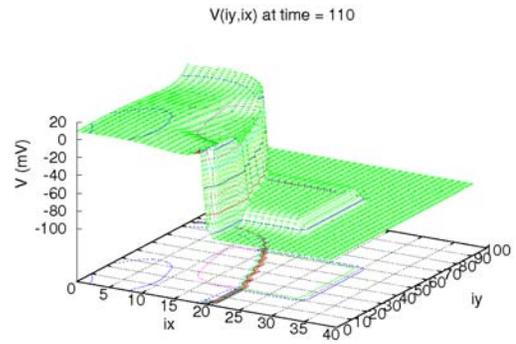
(b) Voltage at $t = 110\,ms$
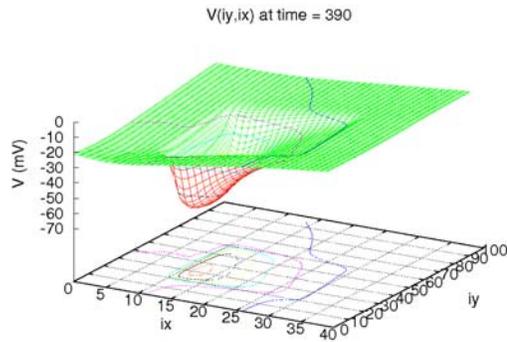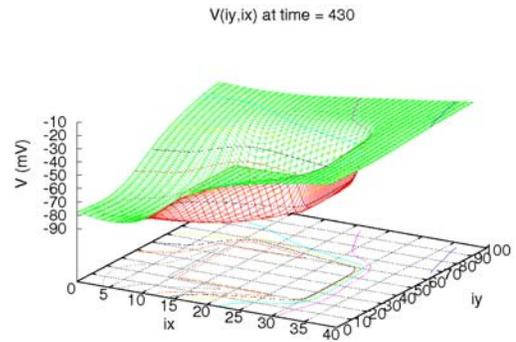
(c) Voltage at $t = 390\,ms$
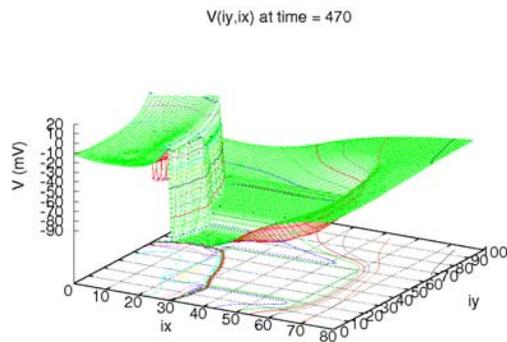
(d) Voltage at $t = 430\,ms$

(e) Voltage at $t = 470\,ms$
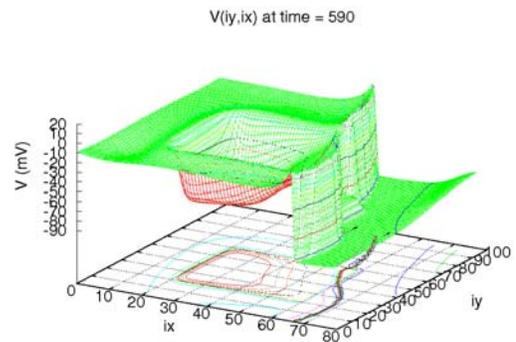
(f) Voltage at $t = 590\,ms$

Figure 5.16: Action potential propagation on tissue of dimensions $51mm \times 31mm$ with abnormal $[K]_0 = 11.0$ in the same rectangular region as described in §5.2.2.

Chapter 6

CONCLUSIONS AND FUTURE WORK

The main contributions of this work are the following:

- Developed serial, space-parallel, and time-parallel implementations of several low and high order, explicit and implicit numerical schemes (Super-Time-Stepping, DuFort-Frankel, RK2imp, RK4, RK45, ...), and compared their performance on the cable equation for problems of various sizes and various parameters. The Super-Time-Stepping scheme turned out to be the most efficient, by far. This is due to the fine spatial discretization required in simulations of propagating action potentials and the very high cost in evaluating the ionic currents (source term); thus, minimizing the number of evaluations, which STS does, proved to be unbeatable.

- Developed the *Extended Parareal Algorithm* (EPA), a new **time-and-space parallel scheme**, which combines the advantages of both space-only and time-only parallelization, to distribute a demanding computational job to multi-processors more efficiently. The performance of the new scheme was thoroughly analyzed and verified computationally on the cable equation problem. Compared to the commonly used space-only parallelization and the standard

Parareal Algorithm for time-only parallelization, the new scheme is highly more scalable and efficient, especially for large scale simulations when large number of processors are available.

- We successfully performed extensive biological simulations of propagating action potentials in one-dimensional and two-dimensional cardiac tissue. This type of simulations could not be achieved before on problems of such large size and fine grids due to the high computational cost. Parallelization, and in particular the new time-and-space parallel scheme, provides a powerful computational method for large scale problems, and it is applicable not only to the problem disscussed in this dissertation, but also potentially to many high computational cost problems arising in other research fields.

This work leads to several future research directions. These include, but not limited to, the following:

1. Extension to three-dimesions,

2. Implementation of other time stepping numerical schemes,

3. Replacing the Luo-Rudy model with other more sophisticated ionic models.

All of these can be easily done in our highly modular program.

Another very interesting research direction is to employ optimal control theory and incorprate it into the framework of the time-and-space parallel scheme to produce optimal electrical defibrillation protocols for control of cardiac arrhythmias. Similar work has been conducted recently by Nagaiah et al. [37, 38]. Due to high computational cost and limitation of serial computing their simulations used the modified FitzHugh-Nagumo model, an overly simplified model with only one gate variable controlling the ionic source, on a single cell (100 μm) for a very short time (4 milliseconds). With the aid of the time-and-space parallel scheme, simulations could be performed with more sophisticated models, such as the Luo-Rudy model or newer, on realistic tissue sizes, to resonably long times.

Bibliography

Bibliography

[1] American Heart Association (AHA). *http://www.heart.org.* 8

[2] F. Alecu. Performance analysis of parallel algorithms. *Journal of applied quantitative methods*, 1, 2007. 44, 46

[3] V. Alexiades, G. Amiez, and P.A. Gremaud. Super-time-stepping acceleration of explicit schemes for parabolic problems. *Commun. Num. Meth. Eng*, 12:12–31, 1996. 24, 26, 27

[4] G. Amdahl. The validity of the single processor approach to achieving large-scale computing capabilities. *In Proceedings of AFIPS Spring Joint Computer Conference*, pages 483–485, 1967. 44, 48

[5] G. Bar. Parallelization in time of (stochastic) ordinary differential equations. *Math. Meth. Anal. Num. (submitted)*, 2002. 62

[6] G. Bar. On the convergence and the stability of the parareal algorithm to solve partial differential equations. *in Proceedings of the 15th international domain decomposition conference*, pages 426–432, 2003. 56, 58

[7] O. Bernus, H. Verschelde, and A.V. Panfilov. Modified ionic models of caridac tissue for efficient large scale computations. *Phys Med Biol.*, 47(11):1947–1959, 2002. 10

[8] P. Bogacki and L.F. Shampine. A 3(2) pair of rungekutta formulas. *Applied Mathematics Letters*, 4(2):321–325, 1989. 37

[9] J.M. Bower and D. Beeman. *The Book of Genesis: Exploring Realistic Neural Models with the GEneral NEural SImulation System. Chapter 4 - The Hodgkin-Huxley Model.* http://www.genesis-sim.org/GENESIS/iBoG/iBoGpdf/, internet edition, 2003. 117

[10] J.W. Buchanan and T Fujino. Ventricular muscle as a functionally continuous medium for electrical propagation: experiments and simulations. *In Imaging, Analysis and Sumulation of the Cardiac System, edited by S. Sideman, R. Beyar. London, Freund Publishing House*, pages 699–717, 1990. 12

[11] J.W. Buchanan and L.S. Gettes. Ionic environment and propagation in cardiac electrophysiology. *In From Cell to Bedside, edited by D.P. Zipes, J. Jalife. Orlando, WB Saunders*, pages 149–156, 1990. 11

[12] J.W. Jr Buchanan, T. Saito, and L.S. Gettes. The effects of antiarrhythmic drugs, stimulation frequency, and potassium induced resting membrane potential changes on conduction velocity and dv/dtmax in guinea pig myocardium. *Circ. Res.*, 56:696–703, 1985. 11

[13] J.R. Cash and A.H. Karp. A variable order runge-kutta method for initial value problems with rapidly varying right-hand sides. *ACM Transactions on Mathematical Software*, 16:201–222, 1990. 37

[14] cellML. luo_rudy_1991_version06. *http://models.cellml.org/luo_rudy_1991_version06.* 20, 119

[15] J.R. Dormand and P.J. Prince. A family of embedded runge-kutta formulae. *Journal of Computational and Applied Mathematics*, 6(1):19–26, 1980. 38

[16] E Fehlberg. Low-order classical runge-kutta formulas with step size control and their application to some heat transfer problems. *NASA Technical Report*, page 315, 1969. 37

108

[17] M.J. Gander and E. Hairer. Nonlinear convergence analysis for the parareal algorithm. *Lecture Notes in Computational Science and Engineering*, 60(1):45–56, 2008. 56, 57

[18] M.J. Gander and S. Vandewalle. Analysis of the parareal time-parallel time-integration method. *SIAM J. Sci. Comput.*, 29(2):556–578, 2007. 56

[19] L.S. Gettes, J.W. Jr Buchanan, T. Saito, Y. Kagiyama, S. Oshita, and T. Fujino. Studies concerned with slow conduction. *In Cardiac Electrophysiology and Arrhythmias, edited by D.P. Zipes, J. Jalife. Orlando, Grune and Stratton*, pages 81–87, 1985. 11

[20] L.S. Gettes, J.L. Hill, and J.W. Buchanan. Future directions of cardiac electrophysiology. *In Progress in Cardiology, vol 10, edited by P.N. Yu, J.F. Goodwin. Philadelphia, Lea and Febiger*, pages 361–371, 1984. 11

[21] GNU. *http://www.gnu.org/software/gsl/*. 38

[22] J.L. Gustafson and E.H. Barsis. Reevaluating amdahl's law. *Communications of the ACM*, 31(5):532–533, 1988. 49

[23] C. R. Harden. Real time computing with the parareal algorithm. *A Masters thesis for the Degree of Master of Science, Florida State University. College of Arts and Sciences*, 2008. 44, 45, 46, 47, 48, 50, 55

[24] M. Hausser. The hodgkin-huxley theory of the action potential. *Nature Neuroscience*, 3:1165, 2000. 16

[25] A.L. Hodgkin and A.F. Huxley. A quantitative description of membrane current and its application to conduction and excitation in nerve. *J. Physiol*, 117(4):500–544, 1952. xii, 4, 16, 17, 18, 116

[26] V. Iyer, R. Mazhari, and R.L. Winslow. A computational model of the human left-ventricular epicaridal myocyte. *Biophys J.*, 87(3):1507–1525, 2004. 10

[27] J. Keener and J. Sneyd. *Mathematical physiology*. Springer, 1998. xii, 12, 13, 15

[28] C. Li and V. Alexiades. Comparison of time stepping schemes on the cable equation. *Electron. J. Diff. Eqns., Conference*, 19:189–196, 2010. 40

[29] C. Li and V. Alexiades. Time stepping for the cable equation, part 1: Serial performance. *Proceedings of Neural, Parallel & Scientific Computations*, 4:241–246, 2010. 40

[30] C. Li and V. Alexiades. Time stepping for the cable equation, part 2: Parallel performance. *Proceedings of Neural, Parallel & Scientific Computations*, 4:247–251, 2010. 51

[31] J.L. Lions, Mayday Y., and Turinici G. A parareal in time discretization of pdes. *C.R. Acad. Sci. Paris Ser. I Math.*, 332:661–668, 2001. 52

[32] C.H. Luo and Y. Rudy. A model of the ventricular cardiac action potential: depolarization, repolarization, and their interaction. *Circ. Res.*, 68:1501–1526, 1991. 4, 10, 19, 20, 119

[33] C.H. Luo and Y. Rudy. A dynamic model of the cardiac ventricular action potential. i. simulations of ionic currents and concentration changes. *Circ. Res.*, 74:1071–1096, 1994. 9, 20

[34] Y. Maday, E.M. Ronquist, and G.A. Staff. The parareal-in-time algorithm: basics, stability analysis and more. *submitted to journal*, 2006. 56, 57, 58

[35] Y. Maday and G. Turinici. The parareal in time iterative solver: A further direction to parallel implementation. *roceedings of the 15th International Domain Decomposition Conference, Lect. Notes Comput. Sci.*, pages 441–448, 2003. 52

[36] D.F. Mayers and K.W. Morton. *Numerical solution of partial differential equations*. Cambridge University Press, 1st edition, 1994. 31

[37] C. Nagaiah, K. Kunisch, and G. Plank. Second order numerical solution for optimal control of monodomain model in cardiac electrophysiology. *Proceedings of ALGORITMY 2009*, pages 202–211, 2009. 105

[38] C. Nagaiah, K. Kunisch, and G. Plank. Numerical solutions for optimal control of monodomain equations in cardiac electrophysiology. *Proceedings of BFG-09, Recent Advances in Optimization and its Applications in Engineering*, 2010. 105

[39] M. Nalik and A.J. Camm. Cardiac electophysiological experiments in numero, part i: Concepts and strategies of mathematical and computer models. *Pacing and Clinical Electrophysiol.*, 14(10):1492–1502, 1991. 2, 3

[40] M. Nalik and A.J. Camm. Cardiac electophysiological experiments in numero, part ii: Models of electrophysiological processes. *Pacing and Clinical Electrophysiol.*, 14(11):1648–1671, 1991. xi, xii, 2, 3, 4, 5

[41] M. Nalik and A.J. Camm. Cardiac electophysiological experiments in numero, part iii: Simulation of arrhythmias and pacing. *Pacing and Clinical Electrophysiol.*, 14(12):2167–2186, 1991. 2, 3

[42] R. Plonsey and R.C. Barr. *Bioelectricity, a quantitative approach.* Springer, 3rd edition, 2007. 12

[43] L. Priebe and D.J. Beuckelmann. Simulation study of cellular electric properties in heart failure. *Circ. Res.*, 82:1206–1223, 1998. 9

[44] W. Rall. *Core conductor theory and cable properties of neurons, in Handbook of Physiology: The Nervous Systme: Cellular Biology of Neurons*. Bethesda, 1977. 13

[45] D. Samaddar, Newman D. E., and Sanchez R. Parallelization in time of numerical simulations of fully- developed plasma turbulence using the parareal algorithm. *Journal of computational physiscs*, 229:6558–6573, 2010. 60

[46] Scholarpedia. *http://en.wikipedia.org/wiki/Heart.* xi, 1

[47] H. Segawa. A family of higher-order implicit time integration methods for unsteady compressible flows. *A dissertation for the Degree of Doctor of Philosophy, North Carolina State University. Department of Aerospace Engineering*, 2011. xii, 39

[48] R.M. Shaw and Y. Rudy. Electrophysiologic effects of acute myocardial ischemia: a theoretical study of altered cell excitability and action potential duration. *Cardiovascular Res.*, 35(2):256–272, 1997. xii, 11

[49] J.R. Sommer and B. Scherer. Geometry of cell and bundle appositions in cardiac muscle: light microscopy. *AM J Physiol.*, 248 (Jeart Circ. Physiol. 17):792–803, 1985. 12

[50] G.A. Staff. The parareal algorithm: A survey of present work. *Technical Report, Norwegian University of Science and Technology. Department of Mathematical Sciences, Norway*, 2003. x, 54, 55, 56, 81

[51] G.A. Staff and E. Ronquist. Stability of the parareal algorithm. *in Proceedings of the 15th International Domain Decomposition Conference, Lecture Notes Computational Science and Engineering*, pages 449–456, 2005. 56, 58, 59

[52] Y. Sun, D. Zhou, A.V. Rangan, and D. Cai. Library-based numerical reduction of the hodgkin-huxley neuron for network simulation. *J. Comput. Neurosci*, 27(3):369–390, 2009. 40

[53] J. Sundnes, G.T. Lines, X. Cai, B.F. Nielsen, K.A. Mardal, and A. Tveito. *Computing the Electrical Activity in the Heart.* Springer, 1st edition, 2006. xi, 2

[54] K.H. Ten Tusscher, O. Bernus, R. Hren, and A.V. Panfilov. Comparison of electrophysiological models for human ventricular cells and tissues. *Progress in Biophysics and Molecular Biology*, 90(1-3):326–345, 2006. 9

[55] K.H. Ten Tusscher, D. Noble, P.J. Noble, and A.V. Panfilov. A model for human ventricular tissue. *Am J Physiol Heart Circ Physiol.*, 286(4):1573–1589, 2004. 10

[56] Cornell University. Online course slides of computing for neurobiology, nionb 441. *http://courses.cit.cornell.edu/bionb441/*, Fall 2006. 7

[57] Wikipedia. *http://en.wikipedia.org/wiki/Action_potential.* xii, 6, 7

[58] Wikipedia. *http://en.wikipedia.org/wiki/Parallel_computing.* xii, 44, 48, 49

[59] Wikipedia. *http://en.wikipedia.org/wiki/Gustafson's_law.* xii, 50

[60] Wikipedia. *http://en.wikipedia.org/wiki/Runge%E2%80%93Kutta_methods.* 35

Appendix

Appendix A

Mathematical Formulation of the Hodgkin-Huxley Model

As mentioned in §2.2.1, the Hodgkin-Huxley model comprises a system of differential equations (2.17)-(2.19). We will present the detailed mathematical formulation of ionic currents here to complete the description of the model.

In the Hodgkin-Huxley model, the total ionic current I_{ion} consists of three ionic currents, all depend on the membrane potential V: a *sodium current* I_{Na}, a *potassium current* I_K, and a *leakage current* I_L, as shown in (2.18). All these three ionic currents are governed by Ohm's law ($I = gV$):

$$I_{Na} = g_{Na}(V - E_{Na})$$

$$I_K = g_K(V - E_K) \tag{A.1}$$

$$I_L = g_L(V - E_L),$$

where E_{ion} is the equilibrium potential (the potential for which the net ionic current flowing across the membrane is zero), a constant provided by experimental data, and g_{ion} is the ionic membrane conductance. Hodgkin and Huxley postulated that g_L is

a constant, whereas g_{Na} and g_K are functions of membrane voltage V to fit their experimental data. The conductances are of the forms:

$$g_{Na} = \overline{g}_{Na} m^3 h,$$

$$g_K = \overline{g}_K n^4, \tag{A.2}$$

$$g_L = \overline{g}_L,$$

where $\overline{g}_{ion}$ is a constant representing the maximal conductance.

m is called "activation gate", h is called "inactivation gate" for sodium, and n is the "activation gate" for potassium. These gate variables, taking values between 0 and 1 and modeling the degree to which corresponding ionic channels are permissive, are governed by first-order ordinary differential equations of the same form:

$$\frac{dm}{dt} = \alpha_m(V)(1 - m) - \beta_m(V)m,$$

$$\frac{dn}{dt} = \alpha_n(V)(1 - n) - \beta_n(V)n, \tag{A.3}$$

$$\frac{dh}{dt} = \alpha_h(V)(1 - h) - \beta_h(V)h,$$

where α's and β's are given by explicit formulas (in units of ms^{-1}, with V in mV) [25]:

$$\alpha_m(V) = 0.1(V + 25) \left/ \left(exp(\frac{V + 25}{10}) - 1 \right) \right. ,$$

$$\beta_m(V) = 4exp(\frac{V}{18}),$$

$$\alpha_h(V) = 0.07exp(\frac{V}{20}),$$

$$\beta_h(V) = 1 \left/ \left(exp(\frac{V + 30}{10}) + 1 \right) \right. ,$$ (A.4)

$$\alpha_n(V) = 0.01(V + 10) \left/ \left(exp(\frac{V + 10}{10}) - 1 \right) \right. ,$$

$$\beta_n(V) = 0.125exp(\frac{V}{80}).$$

A physical interpretation of the gate variables is given in [9]. In the case of an individual ionic channel is considered, each individual gate can be viewed as a probability p_i representing the probability of this gate being in the permissive state. In the case of a large number of ionic channels, p_i can also be interpreted as the fraction of gates that are in the peremissive state and $(1 - p_i)$ as the fraction in the non-permissive state.

Similarly, α's and β's are called *rate constants* ([9]), which govern the rate at which the ion channels transition from non-permissive state to permissive state and vice versa, respectively.

When the membrane potential V is "clamped" at some fixed value, the gates approach their steady state values

$$m^*(V) = \frac{\alpha_m(V)}{\alpha_m(V) + \beta_m(V)},$$

$$n^*(V) = \frac{\alpha_n(V)}{\alpha_n(V) + \beta_n(V)}, \tag{A.5}$$

$$h^*(V) = \frac{\alpha_h(V)}{\alpha_h(V) + \beta_h(V)}.$$

These values are solved from (A.3) by setting the derivatives on the left hand side equal 0. The corresponding time courses for approaching these steady state values are

$$\tau_m(V) = (\alpha_m(V) + \beta_m(V))^{-1},$$

$$\tau_n(V) = (\alpha_n(V) + \beta_n(V))^{-1}, \tag{A.6}$$

$$\tau_h(V) = (\alpha_h(V) + \beta_h(V))^{-1}.$$

Therefore, (A.3) can also be written in the form

$$\frac{dm}{dt} = \frac{1}{\tau_m(V)}(m^*(V) - m),$$

$$\frac{dn}{dt} = \frac{1}{\tau_n(V)}(n^*(V) - n), \tag{A.7}$$

$$\frac{dh}{dt} = \frac{1}{\tau_h(V)}(h^*(V) - h).$$

One can see from (A.7) that the activation and inactivation gates tend to their steady state values at rates governed by the time constants τ_m, τ_n, and τ_h, respectively, which depend purely on the membrane potential V. Thus, activation and inactivation of the ion channels in the Hodgkin-Huxley model do not respond instantaneously to change of membrane potential.

Appendix B

Mathematical Formulation of the Luo-Rudy Phase I (1991) Model

After having a clear view of the Hodgkin-Huxley model, it is much easier to describe the Luo-Rudy 1991 model [32] in the same fasion. In this model, I_{ion} is much more complicated and consists of more ionic currents generated by sodium, potassium and calcium ions

$$I_{ion}(V) = I_{Na}(V) + I_{SI}(V) + I_K(V) + I_{K1(T)}(V). \tag{B.1}$$

The expressions for the ionic currents in equation (B.1) were downloaded from [14] and will be described in detail here.

- I_{Na} is called the *fast sodium current* and defined as

$$I_{Na} = g_{Na} * m^3 * h * j * (V - E_{Na}). \tag{B.2}$$

  The reversal potential E_{Na} is calculated by the Nernst Equation

$$E_{Na} = \frac{RT}{zF} log(\frac{[Na]_o}{[Na]_i}), \tag{B.3}$$

where R is the gas constant, T the temperature, F Faraday's constant, z the valence, $[Na]_o$ the concentration of sodium outside the membrane and $[Na]_i$ is the concentration of sodium inside the membrane. They are all constants. Their values and units are shown in Table B.1.

The activation variable m is governed by the ODE

$$\frac{dm}{dt} = \alpha_m * (1.0 - m) - \beta_m * m, \tag{B.4}$$

where

$$\alpha_m = \frac{0.32 * (V + 47.13)}{1.0 - e^{-0.1*(V+47.13)}} \tag{B.5}$$

and

$$\beta_m = 0.08 * e^{-\frac{V}{11.0}}. \tag{B.6}$$

The fast inactivation variable h is governed by the ODE

$$\frac{dh}{dt} = \alpha_h * (1.0 - h) - \beta_h * h, \tag{B.7}$$

where

$$\alpha_h = \begin{cases} 0.135 * e^{-\frac{80.0+V}{6.8}} & \text{if } V < -40.0 \\ 0.0 & \text{otherwise} \end{cases} \tag{B.8}$$

and

$$\beta_h = \begin{cases} 3.56 * e^{0.079*V} + 310000.0 * e^{0.35*V} & \text{if } V < -40.0 \\ \frac{1.0}{0.13*(1.0+e^{-\frac{V+10.86}{11.1}})} & \text{otherwise} \end{cases} \tag{B.9}$$

The slow inactivation gate j is governed by the ODE

$$\frac{dj}{dt} = \alpha_j * (1.0 - j) - \beta_j * j \tag{B.10}$$

where

$$\alpha_j = \begin{cases} \frac{(-127140.0*e^{0.24444*V} - 0.00003474*e^{-0.04391*V})*(V+37.78)}{1.0+e^{0.311*(V+79.23)}} & \text{if } V < -40.0 \\ 0.0 & \text{otherwise} \end{cases} \quad \text{(B.11)}$$

and

$$\beta_j = \begin{cases} 0.1212 * \frac{e^{-0.01052*V}}{1.0+e^{-0.1378*(V+40.14)}} & \text{if } V < -40.0 \\ 0.3 * \frac{e^{0.0000002535*V}}{1.0+e^{-0.1*(V+32.0)}} & \text{otherwise} \end{cases} \quad \text{(B.12)}$$

- I_{SI} is called the *slow inward current* and defined as

$$I_{SI} = 0.09 * d * f * (V - E_{SI}), \quad \text{(B.13)}$$

where

$$E_{SI} = (7.7 - 13.0287 * log(Cai)). \quad \text{(B.14)}$$

The intracellular calcium concentration Cai is governed by the ODE

$$\frac{dCai}{dt} = -0.0001 * I_{SI} + 0.07 * (0.0001 - Cai). \quad \text{(B.15)}$$

The activation gate d is governed by the ODE

$$\frac{dd}{dt} = \alpha_d * (1.0 - d) - \beta_d * d, \quad \text{(B.16)}$$

where

$$\alpha_d = \frac{0.095 * e^{-0.01*(V-5.0)}}{1.0 + e^{-0.072*(V-5.0)}} \quad \text{(B.17)}$$

and

$$\beta_d = \frac{0.07 * e^{-0.017*(V+44.0)}}{1.0 + e^{0.05*(V+44.0)}}. \quad \text{(B.18)}$$

The inactivation gate f is governed by the ODE

$$\frac{df}{dt} = \alpha_f * (1.0 - f) - \beta_f * f,$$ (B.19)

where

$$\alpha_f = \frac{0.012 * e^{-0.008*(V+28.0)}}{1.0 + e^{0.15*(V+28.0)}}$$ (B.20)

and

$$\beta_f = \frac{0.0065 * e^{-0.02*(V+30.0)}}{1.0 + e^{-0.2*(V+30.0)}}.$$ (B.21)

- I_K is called the *time-dependent potassium current* and defined as

$$I_K = g_K * X * Xi * (V - E_K),$$ (B.22)

where

$$g_K = 0.282 * \sqrt{\frac{[K]_o}{5.4}}$$ (B.23)

with $[K]_o$ being the concentration of potassium ions outside the membrane, which we varied in our experiments for various simulation purposes, and

$$E_K = \frac{R * T}{z * F} * log(\frac{[K]_o + PR\_NAK * [NA]_o}{[K]_i + PR\_NAK * [NA]_i}).$$ (B.24)

The activation gate X is governed by the ODE

$$\frac{dX}{dt} = \alpha_X * (1.0 - X) - \beta_X * X,$$ (B.25)

where

$$\alpha_X = 0.0005 * \frac{e^{0.083*(V+50.0)}}{1.0 + e^{0.057*(V+50.0)}}$$ (B.26)

and

$$\beta_X = 0.0013 * \frac{e^{-0.06*(V+20.0)}}{1.0 + e^{-0.04*(V+20.0)}}.$$ (B.27)

122

The inactivation gate Xi is governed by

$$Xi = \begin{cases} \frac{2.837*e^{0.04*(V+77.0)}-1.0}{(V+77.0)*e^{0.04*(V+35.0)}} & \text{if } V > -100.0 \\ 1.0 & \text{otherwise} \end{cases} \quad \text{(B.28)}$$

- $I_{K1(T)}$ is called the *total time-independent potassium current*, and consists of three components

$$I_{K1(T)} = I_{K1} + I_{Kp} + I_b. \quad \text{(B.29)}$$

The *time-independent potassium current* I_{K1} is governed by

$$I_{K1} = g_{K1} * K1_\infty * (V - E_{K1}), \qquad with \quad K1_\infty = \frac{\alpha_{K1}}{\alpha_{K1} + \beta_{K1}}, \quad \text{(B.30)}$$

where

$$g_{K1} = 0.6047 * \sqrt{\frac{[K]_o}{5.4}}, \quad \text{(B.31)}$$

$$E_{K1} = \frac{R*T}{z*F} * log(\frac{[K]_o}{[K]_i}), \quad \text{(B.32)}$$

$$\alpha_{K1} = \frac{1.02}{1.0 + e^{0.2385*((V-E_{K1})-59.215)}}, \quad \text{(B.33)}$$

and

$$\beta_{K1} = \frac{0.49124 * e^{0.08032*((V+5.476)-E_{K1})} + e^{0.06175*(V-(E_{K1}+594.31))}}{1.0 + e^{-0.5143*((V-E_{K1})+4.753)}}. \quad \text{(B.34)}$$

Unlike $[K]_o$, the concentration of potassium inside the memebrane $[K]_i$ is set to be a constant.

The *plateau potassium current* I_{Kp} is governed by

$$I_{Kp} = g_{KP} * K_P * (V - E_{KP}), \quad \text{(B.35)}$$

where

$$K_p = \frac{1.0}{1.0 + e^{\frac{7.488 - V}{5.98}}} \tag{B.36}$$

and

$$E_{KP} = E_{K1}. \tag{B.37}$$

The *background current* I_b is governed by

$$I_b = g_B * (V - E_B). \tag{B.38}$$

Table B.1: Constants in Luo-Rudy (1991) model

Symbol	Meaning	Unit	value
R	Gas constant	$joules/mole$	8314
T	Temperature	degree	310.0
F	Faraday's constant	$coulombs/mole$	96484.6
g_Na	conductance of sodium	mS/cm^2	23.0
PR_NAK	permeability ratio of Na and K	—	0.01833
g_{KP}	potassium conductance	mS/cm^2	0.0183
E_B	equilibrium current in backgroud	mV	-59.87
g_B	max. leakage conductance	mS/cm^2	0.03921
$[NA]_o$	sodium concentration outside membrane	mM	140.0
$[NA]_i$	sodium concentration inside membrane	mM	18.0
$[K]_i$	potassium concentration inside membrane	mM	145.0
z	Valence	—	1.0

Vita

Chuan Li was born in a small village close to the captail of Sichuan province, China, in 1975. In the fall of 1999, he completed his Bachelors degree in Mathematics with minor in Software Development at the University of Science and Technology of China (USTC), Hefei, Anhui. Then he worked as a software engineer for one year in Beijing Great Dragon Information Technology Co., Ltd. and for two years in Xerox Singapore Software Center, Singapore. In 2003, he came to the United States to continue his studies. In the Summer of 2005, he received his Master degree in Applied Mathematics at Ohio University, Athens OH. Then he transferred to the University of Tennessee, Knoxville (UTK) in pursuit of his Ph.D in the Department of Mathematics. He will complete his studies and receive his Ph.D. in August 2011.

Chuan's research interests are focused on numerical methods for scientific (parallel) computing to model realistic physical processes in scientific and technological problems arising in biology, materials, engineering, and environmental applications.

Chuan will work as a post-doc in a Computational Biopysics & Bioinformatics laboratory in the Department of Physics and Astronomy at Clemson University, under the guidance of Dr. Emil Alexov.