Time-Stepping Methods in Cardiac Electrophysiology

Thomas Roy

Thesis submitted to the Faculty of Graduate and Postdoctoral Studies
in partial fulfillment of the requirements for the degree of Master of Science in
Mathematics [1]

Department of Mathematics and Statistics
Faculty of Science
University of Ottawa

[1]The M.Sc. program is a joint program with Carleton University, administered by the Ottawa-Carleton Institute of Mathematics and Statistics

Abstract

Modelling in cardiac electrophysiology results in a complex system of partial differential equations (PDE) describing the propagation of the electrical wave in the heart muscle coupled with a highly nonlinear system of ordinary differential equations (ODE) describing the ionic activity in the cardiac cells. This system forms the widely accepted bidomain model or its slightly simpler version, the monodomain model. To a large extent, the stiffness of the whole model depends on the choice of the ionic model, which varies in terms of complexity and realism. These simulations require accurate and, depending on the ionic model used, possibly very stable numerical methods. At this time, solving these models numerically requires CPU time of around one day per heartbeat. Therefore, it is necessary to use the most efficient method for these simulations.

This research focuses on the comparison and analysis of several time-stepping methods: explicit or semi-implicit, operator splitting, deferred correction and Rush-Larsen methods. The goal is to find the optimal method for the ionic model used. For our analysis, we used the monodomain model but our results apply to the bidomain model as well. We compare the methods for three ionic models of varying complexity and stiffness: the Mitchell-Schaeffer models with only 2 variables, the more realistic Beeler-Reuter model with 8 variables, and the stiff and very complex ten Tuscher-Noble-Noble-Panfilov (TNNP) models with 17 variables. For each method, we derived absolute stability criteria of the spatially discretized monodomain model and verified

that the theoretical critical time steps obtained closely match the ones in numerical experiments. Convergence tests were also conducted to verify that the numerical methods achieve an optimal order of convergence on the model variables and derived quantities (such as speed of the wave, depolarization time), and this in spite of the local non-differentiability of some of the ionic models. We looked at the efficiency of the different methods by comparing computational times for similar accuracy. Conclusions are drawn on the methods to be used to solve the monodomain model based on the model stiffness and complexity, measured respectively by the most negative eigenvalue of the model's Jacobian and the number of variables, and based on strict stability and accuracy criteria.

Résumé

La modélisation en électrophysiologie cardiaque résulte en un système complexe d'équations aux dérivées partielles (EDP) décrivant la propagation d'une onde électrique dans le muscle cardiaque couplé à un système fortement non-linéaire d'équations différentielles ordinaires (EDO) décrivant l'activité ionique dans les cellules cardiaques. Ce système constitue un modèle largement accepté, le modèle bidomaine, ou sous une version un peu plus simple, le modèle monodomaine. Dans une large mesure, la raideur du modèle dépend du choix du modèle ionique, qui varie en termes de complexité et réalisme. Ces simulations nécessitent des méthodes numériques précises et, selon le modèle ionique utilisé, possiblement très stables. À ce moment, la résolution numérique de ces modèles nécessite des temps de calcul d'environ un jour par battement de cœur. Par conséquent, il est nécessaire d'utiliser les méthodes les plus efficaces pour ces simulations.

Cette recherche se concentre sur la comparaison et l'analyse de plusieurs méthodes itératives en temps: méthodes explicites ou semi-implicites, à pas fractionné, de correction différée et de type exponentielle. Le but est de trouver la méthode optimale pour le modèle ionique utilisé. Pour notre analyse, nous avons utilisé le modèle monodomaine mais nos résultats s'appliquent aussi au modèle bidomaine. Nous comparons les méthodes pour trois modèles ioniques de complexité et de raideur variable: le modèle Mitchell-Schaeffer avec seulement deux variables, le modèle Beeler-Reuter, plus réaliste, avec huit variables, et le modèle ten Tuscher-

Noble-Noble-Panfilov (TNNP), très complexe et raide, avec 17 variables. Pour chaque méthode, nous avons trouvé des critères de stabilité absolue pour le modèle monodomaine discrétisé en espace et vérifié que les pas de temps critiques théoriques correspondent à ceux obtenus dans les expériences numériques. Des tests de convergence ont également été menés pour vérifier que les méthodes numériques atteignent leur ordre optimal de convergence sur les variables du modèle et les autres valeurs dérivés (tels que la vitesse de l'onde, le temps de dépolarisation), et ce en dépit de la non-différentiabilité locale de certains des modèles ioniques. Nous avons examiné l'efficacité des différentes méthodes en comparant les temps de calcul pour une précision similaire. Des conclusions sont tirées sur les méthodes à utiliser pour résoudre le modèle monodomaine, basées sur la rigidité et la complexité du modèle, mesurée respectivement par la valeur propre la plus négative du jacobien du modèle et par rapport à des critères de stabilité et de précision strictes.

Acknowledgements

I would like to express my sincere gratitude to my supervisor Yves Bourgault for all the guidance he provided me throughout my master's degree. I wish to thank him for helping me learn the fundamentals of applied mathematics and for all the time he invested in me. I would also like to thank Yves Coudière and everyone in the CARMEN team at Inria for their warm welcome during my internship in Bordeaux. I also want to thank Charles Pierre for all of his help with the 2D simulations. Last but not least, I want to thank my lovely wife for all of her support.

Dedication

À Gabrielle

Contents

Introduction

The modelling of the electrical activity of the heart offers an interesting perspective on the understanding of cardiac pathologies and its treatments. This subject has great potential in biomedical sciences, as experiments on living hearts require considerable resources. Hence, realistically simulating the behaviour of the heart reduces the necessity for these kinds of experiments. Considering that heart diseases are a leading cause of death in Western countries [5], modelling in cardiac electrophysiology has the potential to significantly impact society. Also, this area has achieved great strides in the past few years, because of improved models, calculation methods and computer power [4, 7, 39].

The mono- and bi- domain models allow for an adequate modelling of the electrical activity in the heart tissue [17, 34]. These models form a complex system of partial differential equations (PDE) that are coupled with a system of ordinary differential equations (ODE) describing the ionic activity in cardiac cells. At this time, solving these models numerically takes a computational time of around one day per heartbeat.

The ODE systems describing ionic activity vary in terms of complexity depending on how realistic one wants the simulation to be. These ionic models, coupled with the mono- or bi- domain model, require discretization in space and time. This thesis focuses on possible types of time-stepping methods: explicit, semi-implicit, operator splitting, exponential and deferred correction methods. The monodomain model is a

standard reaction-diffusion equation and solving the diffusion part of the model explicitly results in severe stability constraints on the time-step. Therefore, the use of semi-implicit or implicit methods is viewed as crucial by most authors [8,18,36]. To achieve this, operator splitting methods have been considered by some authors [13,29,34,37]. Alternatively, an exponential method called the Rush-Larsen (RL) method was created specifically to solve the ODEs resulting from the ionic activity of excitable cells, and it has very stable solutions [28]. This method is very popular and several other variations have been considered, including high-order RL methods [22,24,33].
The different time-stepping methods have been thoroughly studied for non-coupled ionic models [30,31], i.e. only at the cell level, and relatively little at the level of the heart tissue. To date, no research in cardiac electrophysiology has compared a larger number of methods as thoroughly for coupled models. In addition, first-order numerical methods such as Euler's method are still widely used in this field of research due to their easy implementation [25]. We seek to show these are very inefficient compared to high-order methods. Furthermore, we will consider a general ionic model for stability analysis, allowing other researchers to use these results for the model of their choice. Earlier studies are specific to an ionic model, which is limiting in a research area where models are constantly evolving. Thus, this research will provide a summary of the various methods that had not yet been compared.

The goal of this thesis is to find optimal methods for the ionic model used. In the first chapter, we will start by choosing a number of ionic models that cover the range of complexity, stiffness and realism for these types of models.
In the second chapter, we will then list a number of viable numerical methods for the monodomain model. The different methods will be studied with theoretical analysis and numerical tests in the later chapters.
In the third chapter, different stability analysis studies will be conducted for each numerical method. It will be necessary to establish a technique to study the stability

of the numerical methods for the monodomain model in an ODE setting. To ease the comparison from one model to another, a general ionic model will be studied. This analysis will be compared with the numerically observed critical time-step of the different methods.

Then, in the fourth chapter, a convergence test will be performed to check whether the methods of high order exhibit the correct rate of convergence when used to solve the more complex ionic models. The accuracy of the studied methods will then be compared with respect to their accuracy, relative to the size of the time-step and the computational time.

Chapter 1

Models in Cardiac Electrophysiology

In physiology, an action potential (AP) is a short-lasting electrochemical phenomena in which the transmembrane potential of a cell rapidly rises and falls (depolarization and repolarization). At the organ level, this consists of the propagation of a potential wave. We call excitable cells, those that have the ability to transmit action potentials (e.g. cardiac cells, smooth and skeletal muscle cells, secretory cells and neurons). The AP is the result of a transfer of ions into and out of the cell through ion channels in the membrane. To initiate an AP, an electrical stimulus is applied to a cell to trigger a change in the transmembrane potential. If the change is too small, the potential returns almost immediately to its resting value once the applied current is removed. But a larger depolarization, to a certain potential threshold, causes the cell to enter the process of an AP.

In the case of the cardiac AP, this results in a wave of depolarization that propagates though the myocardium, resulting in the contraction of the heart which pumps the blood through the body. We now describe the different phases of a typical ventricular AP (see Figure 1.1).

Phase I. Upstroke phase (fast depolarization): The transmembrane potential is initially at its resting state ($V_{rest} \approx -85\,\mathrm{mV}$), but the partial depolarization from the external stimulus opens the sodium channels and Na^+ ions flood into the cell. This further increases the depolarization such that the cell is now positively charged ($V_{max} \approx 40\,\mathrm{mV}$).

Phase II. Excited phase (plateau): Over a slow time scale, potassium channels open, and K^+ ions flood out of the cell. For a time, Na^+ ions are still flooding in, counterbalancing the outflow of K^+ ions. The transembrane potential falls but very slowly, resulting in a plateau of the cell potential.

Phase III. Downstroke phase (repolarization): The outward potassium current eventually overwhelms the inward sodium current, repolarizing the cell, which closes the sodium channels even more. The cell potential may undershoot its equilibrium value, the cell becoming hyperpolarised.

Phase IV. Recovery phase: Slowly the potassium channels close and the cell returns to its equilibrium state where it can be excited again.

For more information on the cardiac action potential see [3, 17, 34].

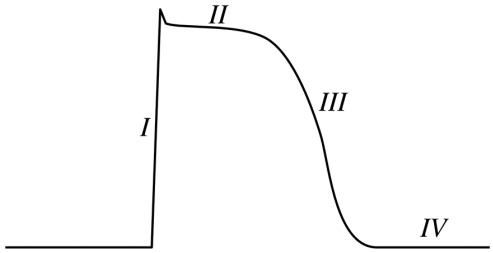

Figure 1.1: The transmembrane potential versus time with the four main phases of the cardiac AP. This is the typical shape of a ventricular AP. (Figure taken from [27])

1.1 Modelling the ionic activity in cells

We present three ionic models that will be used in our numerical test cases. There exist many different models used to describe the ionic activity in cardiac cells. These vary in terms of complexity and stiffness and so we chose to study models that cover the whole spectrum. The first one is a simple model which can be easily understood analytically (e.g. equilibrium points, nullclines, asymptotic solutions, etc.) and is very efficient for numerical simulations. The next two are physiological models which are not easily understood analytically and require more stable numerical methods.

1.1.1 Phenomenological Models

The Mitchell-Schaeffer (MS) model is a two-variable model proposed in [23]. In spite of its very low complexity, this model still accurately represents the main characteristics of the action potential. The original MS model reads:

$$\begin{cases} \dfrac{du}{dt} = f(u,v), \\ \dfrac{dv}{dt} = g(u,v), \end{cases} \tag{1.1.1}$$

where u is the transmembrane potential, v is a gating variable,

$$f(u,v) = -\frac{1}{\tau_{in}} v u^2 (u-1) - \frac{1}{\tau_{out}} u, \quad \text{and} \tag{1.1.2}$$

$$g(u,v) = \begin{cases} \frac{1}{\tau_{open}}(1-v) & \text{for } u < u_{gate}, \\ -\frac{1}{\tau_{close}} v & \text{for } u \geq u_{gate}, \end{cases} \tag{1.1.3}$$

with $\tau_{in} = 0.3$ ms, $\tau_{out} = 6$ ms, $\tau_{open} = 120$ ms, $\tau_{close} = 150$ ms and $u_{gate} = 0.13$. These values of the parameters are proposed in [23].

1.1.2 Physiological Models

A large number of ionic models were developed to represent physiological processes occurring during the action potential. They are written in the following form, where

u is the transmembrane potential, $\mathbf{v}$ is the vector of gating variables and $\mathbf{X}$ is the vector of ionic concentrations.

$$\begin{cases} \dfrac{\partial u}{\partial t} = I(t, u, \mathbf{X}, \mathbf{v}), \\ \dfrac{\partial v_i}{\partial t} = f_i(u, v_i), \quad i = 1, ..., m, \\ \dfrac{\partial \mathbf{X}}{\partial t} = \mathbf{g}(u, \mathbf{X}, \mathbf{v}), \end{cases} \tag{1.1.4}$$

for $t \in (0, T]$ and with initial conditions $u(0) = u^0$, $\mathbf{v}(0) = \mathbf{v}^0$ and $\mathbf{X}(0) = \mathbf{X}^0$. The function I is a source/sink term. In general, the f_i's in (1.1.4) are defined with Hodgkin-Huxley type equations [15]:

$$f_i(u, v_i) = \frac{v_{i,\infty}(u) - v_i}{\tau_{v_i}(u)}, \tag{1.1.5}$$

where $v_{i,\infty}$ is a steady state value for v_i and τ_{v_i} is a time constant, both specific to the ionic model and the gating variable. The steady state values and time constants are dependent on the trans-membrane potential u, whose dependence is usually determined through voltage-clamp experiments [17]. The source/sink term is defined as

$$I(t, u, X, \mathbf{v}) = \frac{1}{C_m}(I_{\text{app}}(t) - I_{\text{ion}}(u, X, \mathbf{v})),$$

where C_m is the membrane capacity, I_{app} is the applied stimulation current, and I_{ion} is the sum of all ionic currents across the cell membrane. The function $\mathbf{g}$ is specific to the ionic model.

The Beeler-Reuter model

One of the ionic models used as part of this research is the Beeler-Reuter (BR) model [2]. For this model, there are six gating variables v_i, denoted x, m, h, j, d and f, and one concentration $[\text{Ca}]_\text{i}$, the intracellular calcium concentration. For each gating variable,

$$\tau_{v_i}(u) = \frac{1}{\alpha_{v_i}(u) + \beta_{v_i}(u)} \quad \text{and} \quad v_{i,\infty}(u) = \frac{\alpha_{v_i}(u)}{\alpha_{v_i}(u) + \beta_{v_i}(u)},$$

Table 1.1: Constants for the rate functions of the BR model

Rate	C_1	C_2	C_3	C_4	C_5	C_6	C_7
α_x	0.0005	0.083	50	0	0	0.057	1
β_x	0.0013	-0.06	20	0	0	-0.04	1
α_m	0	0	47	-1	47	-0.1	-1
β_m	40	-0.056	72	0	0	0	0
α_h	0.126	-0.25	77	0	0	0	0
β_h	1.7	0	22.5	0	0	-0.082	1
α_j	0.055	-0.25	78	0	0	-0.2	1
β_j	0.3	0	32	0	0	-0.1	1
α_d	0.095	-0.01	-5	0	0	-0.072	1
β_d	0.07	-0.017	44	0	0	0.05	1
α_f	0.012	-0.008	28	0	0	0.15	1
β_f	0.0065	-0.02	30	0	0	-0.2	1

where α, β are rate functions defined as

$$\alpha(u) = \frac{C_1 e^{C_2(u+C_3)} + C_4(u+C_5)}{e^{C_6(u+C_3)} + C_7},$$

for each gating variable. The rate constants C_i, $i = 1, ..., 7$, are defined in Table 1.1. The function g in (1.1.4) for the calcium concentration is

$$g(u, \mathbf{v}, [\mathrm{Ca}]_\mathrm{i}) = -10^{-7} \times I_s + 0.07(10^{-7} - [\mathrm{Ca}]_\mathrm{i}). \tag{1.1.6}$$

The source term includes four different ionic currents: an initial fast inward current carried primarily by sodium, I_{Na}; a secondary or slow inward current, I_s, carried mainly but not exclusively by calcium ions; an outward potassium current designated I_{K}; and an outward current designated I_x. They are defined as follows:

$$I_{\mathrm{K}}(u) = 0.35 \left[\frac{4\left(e^{0.04(u+85)} - 1\right)}{e^{0.08(u+53)} + e^{0.04(u+53)}} + 0.2 \frac{u+23}{1 - e^{-0.04(u+23)}} \right], \tag{1.1.7}$$

$$I_x(u,x) = 0.8x\frac{e^{0.04(u+77)}-1}{e^{0.04(u+35)}}, \tag{1.1.8}$$

$$I_{\mathrm{Na}}(u,m,h,j) = (g_{\mathrm{Na}}m^3hj + g_{\mathrm{NaC}})(u - E_{\mathrm{Na}}), \tag{1.1.9}$$

where $g_{\mathrm{Na}} = 4\,\mathrm{mS\,cm^{-2}}$ (S refers to Siemens), $g_{\mathrm{NaC}} = 0.003\,\mathrm{mS\,cm^{-2}}$, and $E_{\mathrm{Na}} = 50\,\mathrm{mV}$, and

$$I_s(u,d,f,[\mathrm{Ca}]_{\mathrm{i}}) = g_s df(u - E_s([\mathrm{Ca}]_{\mathrm{i}})), \tag{1.1.10}$$

where $g_s = 0.009\,\mathrm{mS\,cm^{-2}}$ and

$$E_s([\mathrm{Ca}]_{\mathrm{i}}) = -82.3 - 13.0287\ln[\mathrm{Ca}]_{\mathrm{i}}. \tag{1.1.11}$$

E_{Na} and E_s are the reversal potentials for the corresponding ionic currents.

The ten Tuscher-Noble-Noble-Panfilov Model

A more complex ionic model used in this thesis is the TNNP model from [35]. There exist different versions of this model for different types of cells, and here we consider the epicardial cells. This version is the stiffest of the original TNNP models. This model has seventeen variables, including twelve gating variables and four concentrations. It also has 15 different ionic currents. We present the details of this model in Appendix A.

1.2 Modelling the Heart Tissue

After modelling the electrical activity at the cell level, we now extend our models to the organ level. A good model to describe the electrical activity in the muscle tissue is the widely accepted bidomain model [38]. This continuum-based model uses volume averaging, i.e. the individual cells are not modelled, but instead the models introduced in Section 1.1 are modified to represent local averages of cardiac cells around any location x in the heart. In the bidomain model, the tissue is divided into two superimposed domains representing the intracellular and the extracellular media. See [17, 34] for more details on this model.

1.2.1 The Monodomain Model

The bidomain model is a system of coupled degenerate parabolic equations difficult to solve and analyze. We will consider instead the simpler monodomain model. Assuming equal anisotropy ratio of the extra- and intra-cellular conductances, i.e. $\lambda M_i = M_e$, the system of the bidomain model is reduced to a scalar equation, for the spatial propagation of the transmembrane potential u, namely

$$\frac{\lambda}{1+\lambda}\nabla \cdot (M_i \nabla u) = \chi \left(C_m \frac{\partial u}{\partial t} + I_{\text{ion}} \right), \tag{1.2.1}$$

where M_i is the intracellular conductivity tensor and χ is the cellular membrane area per unit volume of cardiac tissue. The monodomain model is a standard reaction-diffusion equation as opposed to the bidomain model which is a system of two coupled degenerate reaction-diffusion equations. However, the monodomain model does not accurately represent all the features of anisotropic propagation, but this model is realistic enough and much cheaper to solve for many practical applications.

1.2.2 The Coupled Model

For our test cases, we consider the monodomain model with constant conductivities, noted $\sigma_e = \lambda \sigma_i$, coupled with a general ionic cell model. The model then reads as:

$$\frac{\partial u}{\partial t} = \frac{1}{C_m} \left(-I_{ion}(u, \mathbf{v}, \mathbf{X}) + \frac{\lambda}{1+\lambda} \frac{\sigma_i}{\chi} \Delta u \right), \tag{1.2.2}$$

$$\frac{\partial v_i}{\partial t} = f_i(u, v_i) \quad \text{for } i = 1, ...p, \tag{1.2.3}$$

$$\frac{\partial \mathbf{X}}{\partial t} = \mathbf{g}(u, \mathbf{X}, \mathbf{v}), \tag{1.2.4}$$

where σ_i is the intracellular conductivity, u is the transmembrane potential, $\mathbf{v}$ is the vector of gating variables and $\mathbf{X}$ is the vector of ionic concentrations. We will denote $\mathbf{f}$ and $\mathbf{g}$ as the vectors of f_i's and g_i's, respectively. The initial conditions of the problem are given by

$$u(x,0) = u_0(x), \quad \mathbf{v}(x,0) = \mathbf{v}_0(x), \quad \mathbf{X}(x,0) = \mathbf{X}_0(x), \quad x \in \Omega \tag{1.2.5}$$

for some functions u_0, $\mathbf{v}_0$ and $\mathbf{X}_0$, and spatial domain Ω. In our case, we choose these functions as the resting values for each variable, given in Appendix B. We impose homogeneous Neumann boundary conditions, i.e. $\nabla u \cdot n = 0$ on $\partial\Omega$. This amounts to having no current flowing out of the heart.

1.2.3 Nondimensionalization

We nondimensionalize the coupled model (1.2.2)-(1.2.4) as in [27]. This will help with the theoretical analysis in the later sections. We define the following non-dimensional variables and corresponding scales:

$$u = V_m\tilde{u} + V_{rest} \quad \text{where} \quad V_m = V_{max} - V_{rest},$$

$$t = \tilde{t}T, \qquad x = \tilde{x}L,$$

$$X_i = X_{i,m}\tilde{X}_i + X_i^- \quad \text{where} \quad X_{i,m} = X_i^+ - X_i^-, \quad \text{for } i = 1, ..., q,$$

where V_{max}, V_{rest} and V_m are the maximum, resting value and characteristic potentials, respectively. T is the characteristic time and L is the characteristic length. X_i^+, X_i^- and $X_{i,m}$ are the maximum, minimum and characteristic concentrations, respectively. The values for the nondimensionalization scales are given for the MS model in Appendix B. For numerical simulations, we use the standard BR and TNNP models without nondimensionalization.

We obtain the following equations after nondimensionalization

$$\frac{\partial\tilde{u}}{\partial\tilde{t}} = -\tilde{I}_{ion}(\tilde{u}, \mathbf{v}, \tilde{\mathbf{X}}) + \sigma\Delta\tilde{u},$$

$$\frac{\partial\mathbf{v}}{\partial\tilde{t}} = \tilde{\mathbf{f}}(\tilde{u}, \mathbf{v}, \tilde{\mathbf{X}}),$$

$$\frac{\partial\tilde{\mathbf{X}}}{\partial\tilde{t}} = \tilde{\mathbf{g}}(\tilde{u}, \mathbf{v}, \tilde{\mathbf{X}}),$$

where $\sigma = \frac{\lambda}{1+\lambda}\sigma_i T/\chi C_m L^2$,

$$\tilde{I}_{ion}(\tilde{u}, \mathbf{v}, \tilde{\mathbf{X}}) = \frac{T}{C_m V_m} I_{ion}(V_m\tilde{u} + V_{rest}, \mathbf{v}, [X_{1,m}\tilde{X}_1 + X_1^-, ..., X_{q,m}\tilde{X}_q + X_q^-]^T),$$

$$\tilde{\mathbf{f}}(\tilde{u}, \mathbf{v}, \tilde{\mathbf{X}}) = T\mathbf{f}(V_m\tilde{u} + V_{rest}, \mathbf{v}, [X_{1,m}\tilde{X}_1 + X_1^-, ..., X_{q,m}\tilde{X}_q + X_q^-]^T),$$

$$\tilde{g}_i(\tilde{u}, \mathbf{v}, \tilde{\mathbf{X}}) = \frac{T}{X_{i,m}} g_i(V_m\tilde{u}+V_{rest}, \mathbf{v}, [X_{1,m}\tilde{X}_1+X_1^-, ..., X_{q,m}\tilde{X}_q+X_q^-]^T), \quad \text{for } i = 1, ..., q.$$

For simplicity, we drop the tilde in all variables and functions. Our nondimensionalized problem is then: Find u, $\mathbf{v}$ and $\mathbf{X}$ solution of

$$\frac{\partial u}{\partial t} = -I_{ion}(u, \mathbf{v}, \mathbf{X}) + \sigma\Delta u, \tag{1.2.6}$$

$$\frac{\partial \mathbf{v}}{\partial t} = \mathbf{f}(u, \mathbf{v}, \mathbf{X}), \tag{1.2.7}$$

$$\frac{\partial \mathbf{X}}{\partial t} = \mathbf{g}(u, \mathbf{v}, \mathbf{X}). \tag{1.2.8}$$

In some cases, it is simpler to denote $\mathbf{v}$ as the vector of v_i's and X_i's, and $\mathbf{f}$ as the vector of f_i's and g_i's. Equations (1.2.6)-(1.2.8) become:

$$\frac{\partial u}{\partial t} = -I_{ion}(u, \mathbf{v}) + \sigma\Delta u, \tag{1.2.9}$$

$$\frac{\partial \mathbf{v}}{\partial t} = \mathbf{f}(u, \mathbf{v}). \tag{1.2.10}$$

Chapter 2

Numerical Methods

The goal of this chapter is to introduce the different numerical methods studied in the rest of this thesis.

2.1 Concepts and Definitions

This section is a summary of necessary information for the analysis of numerical methods for ODEs, found in [11, 12, 16, 20, 21]. For this section we consider the following initial value problem: Find $y = y(t)$ solution of

$$\begin{cases} \dfrac{dy}{dt} = f(t, y), \quad t \in [0, T], \\ y(0) = y_0. \end{cases} \tag{2.1.1}$$

Theorem 2.1.1. *[20] Let $f(t, y)$ where $f : \mathbb{R} \times \mathbb{R}^m \to \mathbb{R}^m$, be defined and continuous for all (t, y) in the region $D = [0, T] \times \mathbb{R}^m$, and let there exist a constant L such that*

$$\|f(t, y) - f(t, y^*)\| \leq L\|y - y^*\| \tag{2.1.2}$$

holds for every $(t, y), (t, y^*) \in D$. Then for any $y_0 \in \mathbb{R}^m$, there exists a unique solution y of the problem (2.1.1), where $y = y(t)$ is continuous and differentiable with respect to t for all $(t, y) \in D$.

The requirement (2.1.2) is also known as a *Lipschitz condition* and L as a *Lipschitz constant*.

For solving (2.1.1), we will introduce numerical methods to find an approximate solution at a discrete set of nodes, denoted $t_n = n\Delta t$ for $n = 0, 1, ..., N_t = T/\Delta t$. The parameter Δt is called the *time-step* and will be taken as constant. We let y^n denote an approximation to the solution $y(t_n)$ of (2.1.1) at t_n and we want to produce a sequence $\{y^n\}$ approximating the solution of (2.1.1) on the discrete set of points $\{t_n\}$. Such a sequence constitutes a *numerical solution* of the problem.

A *numerical method* is a difference equation involving a number of consecutive approximations y^{n-j} for $j = 0, 1, ..., k$, from which we can sequentially compute the sequence $\{y^n | n = 0, 1, ..., N_t\}$. If $k = 0$, the method is called a *one-step* method while if $k > 0$, the method is called a *multistep* or $(k+1)$-*step* method. A $(k+1)$-step method requires $k + 1$ starting values before the sequence $\{y^n\}$ can be computed. Finding such additional starting values usually requires an appropriate one-step method, but some multistep methods are self-starting and do not require another method. This will be discussed at the end of this chapter.

Almost all of the methods introduced in this chapter can be written in the form

$$y^{n+1} = \sum_{j=0}^{k} a_j y^{n-j} + \Delta t \sum_{j=-1}^{k} b_j f(t_{n-j}, y^{n-j}), \quad n \geq k, \tag{2.1.3}$$

where the coefficients $a_0, ..., a_k, b_{-1}, b_0, ..., b_k$ are constants and $k \geq 0$. Assuming $|a_k| + |b_k| \neq 0$, it is a $(k + 1)$-step method. If $b_{-1} = 0$, the method is said to be *explicit*, while if $b_{-1} \neq 0$, it is said to be *implicit*. The function $f(t, y)$ can be written as $f(t, y) = g(t, y) + h(t, y)$. If the method takes g explicitly and h implicitly, it is said to be *semi-implicit* (*IMEX, implicit-explicit*).

We consider the numerical solution of the initial value problem (2.1.1) given by the general method (2.1.3) with appropriate $k + 1$ starting values α_n, that is the solution

given by

$$\begin{cases} y^{n+1} = \sum_{j=0}^{k} a_j y^{n-j} + \Delta t \sum_{j=-1}^{k} b_j f(t_{n-j}, y^{n-j}), \quad n \geq k, \\ y^n = \alpha_n, \quad n = 0, 1, ..., k. \end{cases} \tag{2.1.4}$$

Definition 2.1.2. *The method defined in (2.1.4) is said to be* convergent *if, for all initial value problems satisfying the hypotheses of Theorem 2.1.1, we have that*

$$\max_{0 \leq n \leq N_t} \|y(t_n) - y^n\| \to 0 \quad \textit{as } \Delta t \to 0.$$

Such convergence is not easily proven from the definition and thus we will need to define some conditions under which a method is convergent. One such condition is that it has to be a sufficiently accurate representation of the differential system. As a measure of accuracy, we define the *local truncation error* by

$$T_n(y) = y(t_n) - \left[\sum_{j=0}^{k} a_j y(t_{n-j}) + \Delta t \sum_{j=-1}^{k} b_j f(t_{n-j}, y(t_{n-j})) \right], \tag{2.1.5}$$

for $n \geq k$. Some authors define the local truncation error by the function $\tau_n(y) = \frac{1}{\Delta t} T_n(y)$.

Definition 2.1.3. *The method (2.1.4) is said to be* consistent *if, for all initial value problems satisfying the hypotheses of Theorem 2.1.1*

$$\tau(\Delta t) \equiv \max_{0 \leq t_n \leq T} |\tau_n(y)| \to 0 \quad \textit{as } \Delta t \to 0. \tag{2.1.6}$$

A method is said of *order* p if $\tau(\Delta t) = \mathcal{O}(\Delta t^p)$. If a method satisfies the consistency condition (2.1.6), then it has the appropriate accuracy necessary for convergence. However, consistency alone is not sufficient. Stability is required.
When considering the stability of the problem (2.1.1), we can perturb both the function f and initial condition y_0 and study how sensitive the solution is to such perturbations. The perturbation $(\delta(t), \delta_0)$ and the perturbed solution $z(t)$ are defined by

the perturbed initial value problem

$$\begin{cases} \dfrac{dz}{dt} = f(t,y) + \delta(x), \quad t \in [0,T], \\ y(0) = y_0 + \delta_0. \end{cases} \tag{2.1.7}$$

A numerical method applied to (2.1.1) introduces errors due to discretization or round-off, and these errors could be interpreted as perturbations to the problem (2.1.4). The perturbation $\{(\delta^n, n = 0, 1, ...N_t\}$ and the perturbed solution $\{z^n, n = 0, 1, ...N_t\}$ of (2.1.4) are defined by the perturbed difference system

$$\begin{cases} z^{n+1} = \sum_{j=0}^{k} a_j z^{n-j} + \Delta t \left(\sum_{j=-1}^{k} b_j f(t_{n-j}, z^{n-j}) + \delta^{n+1} \right), \quad n \geq k, \\ z^n = \alpha_n + \delta_0^n, \quad n = 0, 1, ..., k. \end{cases} \tag{2.1.8}$$

Definition 2.1.4. *Let $\{\delta^n, n = 0, 1, ...N_t\}$ and $\{\delta^{n,*}, n = 0, 1, ...N_t\}$ be any two perturbations of (2.1.4) and let $\{z^n, n = 0, 1, ...N_t\}$ and $\{z^{n,*}, n = 0, 1, ...N_t\}$ be the resulting perturbed solutions. Then if there exist constants S and Δt_0 such that, for all $\Delta t \in (0, \Delta t_0]$,*

$$\|z^n - z^{n,*}\| \leq S\epsilon \text{ for } 0 \leq n \leq N_t \quad \text{whenever} \quad \|\delta^n - \delta^{n,*}\| \leq \epsilon \text{ for } 0 \leq n \leq N_t, \tag{2.1.9}$$

we say that the method (2.1.3) is zero-stable.

The stability of (2.1.3) is also linked to the roots of the polynomial

$$\rho(\zeta) = \zeta^{k+1} - \sum_{j=0}^{k} a_j \zeta^{k-j}. \tag{2.1.10}$$

Definition 2.1.5. *The method (2.1.3) is said to satisfy the* root condition *if all roots of $\rho(\zeta)$ have modulus less than or equal to unity, and those of modulus unity are simple.*

Theorem 2.1.6. *[20] A method of the form (2.1.3) satisfying the root condition is zero-stable.*

Theorem 2.1.7. *[20] The necessary and sufficient conditions for the method (2.1.3) to be convergent is that it be both consistent and zero-stable.*

We now introduce another important notion for the stability of numerical methods which will be used in Chapter 3. Zero-stability is defined for very small time-steps, i.e. $\Delta t \to 0$, and so does not consider the impact of the function $f(t, y)$ on the stability of the method. In fact, it only considers the problem $y' = 0$. We need a definition of stability that gives a condition on the largest possible size of the time-step used, depending on the function $f(t, y)$. We first consider the following initial value problem

$$\begin{cases} \dfrac{dy}{dt} = \lambda y, \\ y(0) = 1, \end{cases} \tag{2.1.11}$$

the famous *Dahlquist test equation.* For one-step methods of the form (2.1.3), we can always find a function $R(z)$ such that the method applied to (2.1.11) may be written as

$$y^{n+1} = R(z)y^n, \tag{2.1.12}$$

where $z = \Delta t\ \lambda$. For the Euler's method, we have $R(z) = 1 + z$.

Definition 2.1.8. *The function $R(z)$ is called the* stability function *of the method. It can be interpreted as the numerical solution after one step for the Dahlquist test equation. The set*

$$S = \{z \in \mathbb{C}; |R(z)| \leq 1\} \tag{2.1.13}$$

is called the stability domain *or* stability region *or* region of absolute stability *of the method, and the method is said to be* absolutely stable.

Definition 2.1.9. *A method, whose stability domain satisfies*

$$S \supset \mathbb{C}^- = \{z;\ Re(z) \leq 0\},$$

is called A-stable.

We now define these concepts in the case of multi-step methods. We consider a general $(k+1)$-step method of the form (2.1.3) applied to the Dahlquist test equation and rewrite it in this form:

$$(1 - zb_{-1})y^{n+1} + (a_0 - zb_0)y^n + ... + (a_k - zb_k)y^{n-k} = 0, \quad z = \Delta t\ \lambda \tag{2.1.14}$$

Equation (2.1.14) is solved using Lagrange's method found in [11]. We set $y^j = \zeta^j$ and divide by ζ^{n-k} to obtain the characteristic equation

$$(1 - zb_{-1})\zeta^{k+1} + (a_0 - zb_0)\zeta^k + ... + (a_k - zb_k) = 0. \tag{2.1.15}$$

The difference equation (2.1.14) has stable solutions (for arbitrary starting values) iff all the roots of (2.1.15) have modulus less than or equal to unity and multiple roots must have modulus strictly smaller than 1.

Definition 2.1.10. *The set*

$$S = \left\{ z \in \mathbb{C}\ ; \begin{array}{l} \textit{all roots } \zeta_j(z) \textit{ of (2.1.15) satisfy } |\zeta_j(z)| \leq 1, \\ \textit{multiple roots satisfy } |\zeta_j(z)| < 1 \end{array} \right\} \tag{2.1.16}$$

is the stability domain *or* stability region *or* region of absolute stability *of the method. We have* A-stability *if* $S \supset \mathbb{C}^-$.

This concept of absolute stability can be extended beyond the scalar case. Consider a linear system

$$\frac{dy}{dt} = Ay, \tag{2.1.17}$$

where A is a constant $m \times m$ matrix. For simplicity, we suppose that A is diagonalizable, which means it has a set of m linearly independent eigenvectors v_p such that $Av_p = \lambda_p v_p$ for $p = 1, ..., m$, where λ_p are the corresponding eigenvalues. Let $P = [v_1, ..., v_m]$ be the matrix of eigenvectors and $D = \text{diag}(\lambda_1, ..., \lambda_m)$ be the diagonal matrix of eigenvalues, then

$$A = PDP^{-1} \quad \text{and} \quad D = P^{-1}AP.$$

Let $u(t) = P^{-1}y(t)$. We can rewrite (2.1.17) as

$$\frac{du}{dt} = Du. \tag{2.1.18}$$

This is a diagonal system of equations that we decouple into m independent scalar equations of the form

$$\frac{du_p}{dt} = \lambda_p u_p, \quad \text{for } p = 1, ...m, \tag{2.1.19}$$

where u_p are the components of u. A linear multistep method of the form (2.1.3) applied to (2.1.17) can also be decoupled in the same way. For the overall method to be stable, each of the scalar problems must be stable, and this requires $\Delta t\lambda_p$ to be in the stability region of the method for $p = 1, ..., m$. This can be rewritten as a condition on the spectral radius of the matrix A, $\rho(A)$.

The concept of absolute stability does not directly apply to nonlinear systems. As in [12], we will consider a linearized approximation of the nonlinear system.

$$\frac{dy}{dt} = f(t, y). \tag{2.1.20}$$

Let $\varphi(t)$ be a smooth solution of (2.1.20). We linearize f around $\varphi(t)$ as follows

$$y'(t) = f(t, \varphi(t)) + \frac{\partial f}{\partial y}(t, \varphi(t))(y(t) - \varphi(t)) + \mathcal{O}(\|y - \varphi\|^2). \tag{2.1.21}$$

We let $u(t) = y(t) - \varphi(t)$ to obtain

$$u'(t) = \frac{\partial f}{\partial y}(t, \varphi(t))u(t) + \mathcal{O}(\|u\|^2) = J(t)u(t) + \mathcal{O}(\|u\|^2), \tag{2.1.22}$$

where $J(t)$ is the Jacobian matrix of the system. As an approximation, we consider the Jacobian to be constant and drop $\mathcal{O}(\|u\|^2)$ to obtain the linear system

$$\frac{du}{dt} = Ju. \tag{2.1.23}$$

The stability analysis that we presented for linear systems can now be used on this linearized approximation.

Stiffness

The stiffness of an initial value problem arises when solving it numerically. When a problem is stiff, the main restriction on the time-step used for the numerical method is due to its stability rather than for accuracy requirements. In the case of a nonlinear system of ODEs, the eigenvalues of the Jacobian determine the stability of the problem. In this thesis, we will say that a system of ODEs is "stiff" if it has an eigenvalue with a large negative real part, thus requiring a very small Δt for $\Delta t\lambda$ to be in the stability region of the numerical method used. This differs from the usual definition which uses the stiffness ratio [20]. Stiff problems tend to be solved using implicit methods because of their large stability region. The BR and TNNP models are considered stiff, but not the MS model.

2.2 Space discretization

The problem (1.2.6)-(1.2.8) has derivatives with respect to time and space and thus requires both spatial and temporal discretization. Depending on the spatial dimension of the problem, we consider the following methods of spatial discretization.

2.2.1 Finite difference method

We consider the 1D version of the problem (1.2.6)-(1.2.8). We consider evenly spaced grid points $x_0, ..., x_N$ covering our spatial domain $[0, L]$, with $h = L/N$, $x_j = jh$ for each $j = 0, 1, ..., N$. We denote $U = [U_0, ..., U_N]^T \simeq [u(x_0, t), ..., u(x_N, t)]^T$, $V = [V_0, ..., V_N]^T \simeq [v(x_0, t), ..., v(x_N, t)]^T$, where $v(x, t) = [v_1(x, t), ..., v_p(x, t)]^T$ and $X = [X_0, ..., X_N]^T \simeq [X(x_0, t), ..., X(x_N, t)]^T$, where $X(x, t) = [X_1(x, t), ..., X_q(x, t)]^T$. The second derivative in space is approximated as:

$$\frac{d^2U_i}{dx^2} \approx \frac{U_{i+1} - 2U_i + U_{i-1}}{h^2} \quad \text{for } i = 1, ..., N-1. \tag{2.2.1}$$

Discretizing in space, our model becomes

$$U_t = -I(U, V, X) + AU, \tag{2.2.2}$$

$$V_t = F(U, V, X), \tag{2.2.3}$$

$$X_t = G(U, V, X), \tag{2.2.4}$$

where I, F and G are the vectors of I_{ion}, $\mathbf{f}$ and $\mathbf{g}$ evaluated at the solution at each spatial node. A is the $(N+1) \times (N+1)$ diffusion matrix defined as

$$A = \frac{\sigma}{h^2} \begin{bmatrix} -1 & 1 & 0 & \cdots & 0 \\ 1 & -2 & 1 & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & -2 & 1 \\ 0 & \cdots & 0 & 1 & -1 \end{bmatrix}.$$

This matrix is derived from (2.2.1) using homogeneous Neumann boundary conditions, i.e. $\frac{du}{dx}(0,t) = \frac{du}{dx}(L,t) = 0$. Indeed, using forward difference at $x = 0$ we get $\frac{U_1 - U_0}{h} = 0$ and using backward difference at $x = L$, $\frac{U_N - U_{N-1}}{h} = 0$, which we both multiply by σ/h before introducing these in the matrix A. This gives $U_0 = U_1$ and $U_{N-1} = U_N$.

2.2.2 Finite element method

We also consider the 2D version of the problem (1.2.6)-(1.2.8) using the finite element method [6]. Here, we assume that our spatial domain Ω can be covered by a regular mesh $\mathscr{T}_h$ of triangles of maximum diameter h, with $N+1$ nodes, noted $x_0, ..., x_N$. We denote V_h as the space of continuous linear finite elements on $\mathscr{T}_h$ [26] and consider the usual basis of hat function $\Phi_0^h, ..., \Phi_N^h$ corresponding to each node $x_0, ..., x_N$, respectively. We again suppose homogeneous Neumann boundary conditions for u on $\partial\Omega$. We set $u^h(t) = \sum_{j=0}^{N} U_j \Phi_j^h$, $v_i^h(t) = \sum_{j=0}^{N} V_{i,j} \Phi_j^h$ for $i = 1, ..., p$ and

$X_i^h(t) = \sum_{j=0}^{N} X_{i,j}\Phi_j^h$ for $i = 1, ..., q$.

The variational formulation of the problem reads as: Find $(u^h, \mathbf{v}^h, \mathbf{X}^h) \in \mathscr{C}([0,T], V_h)^{p+q+1}$ solution of

$$\int_\Omega u_t^h \Phi_i^h dx = -\int_\Omega I_{ion}(u^h, \mathbf{v}^h, \mathbf{X}^h)\Phi_i^h dx - \int_\Omega \sigma \nabla u^h \nabla \Phi_i^h dx, \tag{2.2.5}$$

$$\int_\Omega v_{k,t}^h \Phi_i^h dx = \int_\Omega f_k(u^h, \mathbf{v}^h, \mathbf{X}^h)\Phi_i^h dx, \quad \text{for } k = 1, ...p, \tag{2.2.6}$$

$$\int_\Omega X_{l,t}^h \Phi_i^h dx = \int_\Omega g_l(u^h, \mathbf{v}^h, \mathbf{X}^h)\Phi_i^h dx, \quad \text{for } l = 1, ...q, \tag{2.2.7}$$

for all $i = 0, ..., N$. The integrals involving nonlinear functions are not easily computed, therefore we approximate them using *nonlinear mass-lumping*, i.e.

$$\int_\Omega I_{ion}(u^h, v^h, X^h)\Phi_i^h dx \approx \int_\Omega \sum_{j=0}^{N} I_{ion}(U_j, V_j, X_j)\Phi_j^h \Phi_i^h dx,$$

and similarly for the integrals in (2.2.6)-(2.2.7). We then rewrite (2.2.5)-(2.2.7) under matrix form:

$$MU_t = -MI(U, V, X) + AU \tag{2.2.8}$$

$$MV_t = MF(U, V, X), \tag{2.2.9}$$

$$MX_t = MG(U, V, X), \tag{2.2.10}$$

where U, V, X, I, F and G are defined as in Section 2.2.1, A is the stiffness matrix obtained by integrating by parts the second order term, M is the mass matrix such that $M_{ij} = \int_\Omega \Phi_i^h \Phi_j^h dx$, or its approriate modification to handle the vectorial variables V and X.

2.3 Time discretization

We consider a variety of explicit, semi-implicit and implicit methods of first to third-order, using a constant time-step Δt, for the time discretization. When solving

reaction-diffusion equations as the monodomain model, some authors suggest using semi-implicit methods [8,24,34]. Implicit methods are usually too expensive for many ionic models and as we will see in the next section, explicit methods have additional stability restrictions.

Let us introduce the notation for the fully discretized version of the problem (1.2.6)-(1.2.8). We denote

$$U^n = [U_0^n, ..., U_{N_t}^n]^T \simeq [u(x_0, t_n), ..., u(x_{N_t}, t_n)]^T,$$

$$V^n = [V_{1,0}^n, ..., V_{p,0}^n, ..., V_{p,N_t}^n]^T \simeq [v_1(x_0, t_n), ..., v_p(x_0, t_n), ..., v_p(x_{N_t}, t_n)]^T \text{ and}$$

$$X^n = [X_{1,0}^n, ..., X_{q,0}^n, ..., X_{q,N_t}^n]^T \simeq [X_1(x_0, t_n), ..., X_q(x_0, t_n), ..., X_q(x_{N_t}, t_n)]^T.$$

We define the methods for the finite difference method. Defining the methods for the finite element method only requires adding the mass matrix M to the left-hand-side of the equations and in front of the I, F and G.

For most methods, the gating variables and concentrations are treated the same way. Therefore, if the following schemes do not have X and G, it is implied they are included in V and F, respectively. We first briefly present all the numerical schemes and cover their details afterwards.

2.3.1 First order methods

(i) Forward Euler (FE):

$$\begin{aligned} \frac{U^{n+1} - U^n}{\Delta t} &= -I(U^n, V^n) + AU^n, \\ \frac{V^{n+1} - V^n}{\Delta t} &= F(U^n, V^n). \end{aligned} \tag{2.3.1}$$

(ii) Forward-Backward Euler (FBE):

$$\begin{aligned} \frac{U^{n+1} - U^n}{\Delta t} &= -I(U^n, V^n) + AU^{n+1}, \\ \frac{V^{n+1} - V^n}{\Delta t} &= F(U^n, V^n). \end{aligned} \tag{2.3.2}$$

(iii) Rush-Larsen ($c_{-1} = 0$, $c_0 = 1$, $c_1 = 0$) with Forward-Backward Euler (RL-FBE):

$$\begin{aligned} \frac{U^{n+1} - U^n}{\Delta t} &= -I(U^n, V^n, X^n) + AU^{n+1}, \\ \frac{V_i^{n+1} - V_i^n}{\Delta t} &= \Phi(a_i^n \Delta t)(a_i^n V_i^n + b_i^n), \quad i = 1, ..., p, \\ \frac{X^{n+1} - X^n}{\Delta t} &= G(U^n, V^n, X^n), \end{aligned} \tag{2.3.3}$$

where Φ, a_i^n and b_i^n are given below.

2.3.2 Second order methods

(i) Second order semi-implicit backward differentiation (SBDF2):

$$\begin{aligned} \frac{\frac{3}{2}U^{n+1} - 2U^n + \frac{1}{2}U^{n-1}}{\Delta t} &= -2I(U^n, V^n) + I(U^{n-1}, V^{n-1}) + AU^{n+1}, \\ \frac{\frac{3}{2}V^{n+1} - 2V^n + \frac{1}{2}V^{n-1}}{\Delta t} &= 2F(U^n, V^n) - F(U^{n-1}, V^{n-1}). \end{aligned} \tag{2.3.4}$$

(ii) Strang Splitting with Crank-Nicolson and Runge-Kutta 4 (CN-RK4):

Here we denote $Y^n = \begin{bmatrix} U^n \\ V^n \end{bmatrix}$, $Y^* = \begin{bmatrix} U^* \\ V^* \end{bmatrix}$ and $Y^{**} = \begin{bmatrix} U^{**} \\ V^* \end{bmatrix}$.

Step 1

$$\frac{Y^* - Y^n}{\Delta t/2} = \frac{1}{6}(K_1 + 2K_2 + 2K_3 + K_4), \tag{2.3.5}$$

where

$$K_1 = \begin{bmatrix} -I(Y^n) \\ F(Y^n) \end{bmatrix}, \quad K_2 = \begin{bmatrix} -I(Y^n + \frac{\Delta t}{4}K_1) \\ F(Y^n + \frac{\Delta t}{4}K_1) \end{bmatrix},$$

$$K_3 = \begin{bmatrix} -I(Y^n + \frac{\Delta t}{4}K_2) \\ F(Y^n + \frac{\Delta t}{4}K_2) \end{bmatrix}, \quad K_4 = \begin{bmatrix} -I(Y^n + \frac{\Delta t}{2}K_3) \\ F(Y^n + \frac{\Delta t}{2}K_3) \end{bmatrix}.$$

Step 2

$$\frac{U^{**} - U^*}{\Delta t} = \frac{1}{2}A(U^{**} + U^*). \tag{2.3.6}$$

Step 3

$$\frac{Y^{n+1} - Y^{**}}{\Delta t/2} = \frac{1}{6}(K_1^* + 2K_2^* + 2K_3^* + K_4^*), \tag{2.3.7}$$

where

$$K_1^* = \begin{bmatrix} -I(Y^{**}) \\ F(Y^{**}) \end{bmatrix}, \quad K_2^* = \begin{bmatrix} -I(Y^{**} + \frac{\Delta t}{4} K_1^*) \\ F(Y^{**} + \frac{\Delta t}{4} K_1^*) \end{bmatrix},$$

$$K_3^* = \begin{bmatrix} -I(Y^{**} + \frac{\Delta t}{4} K_2^*) \\ F(Y^{**} + \frac{\Delta t}{4} K_2^*) \end{bmatrix}, \quad K_4^* = \begin{bmatrix} -I(Y^{**} + \frac{\Delta t}{2} K_3^*) \\ F(Y^{**} + \frac{\Delta t}{2} K_3^*) \end{bmatrix}.$$

(iii) Strang Splitting with Crank-Nicolson and Runge-Kutta 2 (CN-RK2):

Here we denote $Y^n = \begin{bmatrix} U^n \\ V^n \end{bmatrix}$, $Y^* = \begin{bmatrix} U^* \\ V^* \end{bmatrix}$ and $Y^{**} = \begin{bmatrix} U^{**} \\ V^* \end{bmatrix}$.

Step 1

$$\frac{Y^* - Y^n}{\Delta t/2} = K_2, \tag{2.3.8}$$

where K_2 is defined as in Step 1 of the CN-RK4 splitting scheme.

Step 2

$$\frac{U^{**} - U^*}{\Delta t} = \frac{1}{2} A(U^{**} + U^*). \tag{2.3.9}$$

Step 3

$$\frac{Y^{n+1} - Y^{**}}{\Delta t/2} = K_2^*, \tag{2.3.10}$$

where K_2^* is defined as in Step 3 of the CN-RK4 splitting scheme.

(iv) Second Order Rush-Larsen ($c_{-1} = 0$, $c_0 = \frac{3}{2}$, $c_1 = -\frac{1}{2}$) with Crank-Nicolson Adam-Bashforth (RL-CNAB):

$$\begin{aligned} \frac{U^{n+1} - U^n}{\Delta t} &= -\frac{3}{2} I(U^n, V^n, X^n) + \frac{1}{2} I(U^{n-1}, V^{n-1}, X^{n-1}) + \frac{1}{2} A(U^{n+1} + U^n), \\ \frac{V_i^{n+1} - V_i^n}{\Delta t} &= \Phi(a_i^{n+\frac{1}{2}} \Delta t)(a_i^{n+\frac{1}{2}} V_i^n + b_i^{n+\frac{1}{2}}), \quad i = 1, ..., p, \\ \frac{X^{n+1} - X^n}{\Delta t} &= \frac{3}{2} G(U^n, V^n, X^n) - \frac{1}{2} G(U^{n-1}, V^{n-1}, X^{n-1}), \end{aligned} \tag{2.3.11}$$

where Φ, a_i^n and b_i^n are given below.

2.3.3 Third order methods

(i) Third Order Deferred Correction (DC3):

Here we denote $Y^n = \begin{bmatrix} U^n \\ V^n \end{bmatrix}$ and $Y_i^n = \begin{bmatrix} U_i^n \\ V_i^n \end{bmatrix}$. The initial values for the algorithm are $Y_0^0 = Y_0$, $Y_1^0 = 0$ and $Y_2^0 = 0$.

$$\text{for } n \geq 0 \quad \begin{cases} ml_0^{n+1} = -I(Y_0^n), \quad nl_0^{n+1} = F(Y_0^n), \\ \dfrac{U_0^{n+1} - U_0^n}{\Delta t} = ml_0^{n+1} + AU_0^{n+1}, \quad \dfrac{V_0^{n+1} - V_0^n}{\Delta t} = nl_0^{n+1}, \\ dU_0^{n+1} = (U_0^{n+1} - U_0^n)/\Delta t, \quad dV_0^{n+1} = (V_0^{n+1} - V_0^n)/\Delta t, \end{cases} \tag{2.3.12}$$

$$\text{for } n \geq 1 \quad \begin{cases} d^2U_0^{n+1} = (dU_0^{n+1} - dU_0^n)/\Delta t, \quad ml_1^n = F(Y_0^n + \Delta t Y_1^{n-1}), \\ \dfrac{U_1^n - U_1^{n-1}}{\Delta t} = AU_1^n - \dfrac{1}{2} d^2U_0^{n+1} + \dfrac{ml_1^n - ml_0^n}{\Delta t}, \\ dU_1^n = (U_1^n - U_0^{n-1})/\Delta t, \\ d^2V_0^{n+1} = (dV_0^{n+1} - dV_0^n)/\Delta t, \quad nl_1^n = F(Y_0^n + \Delta t Y_1^{n-1}), \\ \dfrac{V_1^n - V_1^{n-1}}{\Delta t} = -\dfrac{1}{2} d^2V_0^{n+1} + \dfrac{nl_1^n - nl_0^n}{\Delta t}, \\ dV_1^n = (V_1^n - V_0^{n-1})/\Delta t, \end{cases} \tag{2.3.13}$$

$$\text{for } n \geq 2 \quad \begin{cases} d^2U_1^n = (dU_1^n - dU_1^{n-1})/\Delta t \quad d^3U_0^{n+1} = (d^2U_0^{n+1} - d^2U_0^n)/\Delta t, \\ ml_2^{n-1} = F(Y_0^{n-1} + \Delta t Y_1^{n-1} + \Delta t^2 Y_2^{n-2}), \\ \dfrac{U_2^{n-1} - U_1^{n-2}}{\Delta t} = AU_2^{n-1} - \dfrac{1}{2}d^2U_1^n + \dfrac{1}{6}d^3U_0^n + \dfrac{ml_2^{n-1} - ml_1^{n-1}}{\Delta t^2}, \\ d^2V_1^n = (dV_1^n - dV_1^{n-1})/\Delta t \quad d^3V_0^{n+1} = (d^2V_0^{n+1} - d^2V_0^n)/\Delta t, \\ nl_2^{n-1} = F(Y_0^{n-1} + \Delta t Y_1^{n-1} + \Delta t^2 Y_2^{n-2}), \\ \dfrac{V_2^{n-1} - V_1^{n-2}}{\Delta t} = -\dfrac{1}{2}d^2V_1^n + \dfrac{1}{6}d^3V_0^n + \dfrac{nl_2^{n-1} - nl_1^{n-1}}{\Delta t^2}, \\ Y^{n-1} = Y_0^{n-1} + \Delta t Y_1^{n-1} + \Delta t^2 Y_2^{n-1}. \end{cases} \tag{2.3.14}$$

We now make comments and explain some details for the methods above.

Euler's Methods

Euler's method is a very well known basic time-stepping method. We solve our system using the explicit Forward Euler's method but we can choose to take the diffusion term implicitly to get the semi-implit Forward-Backward Euler's method. As we will see in the next chapter, solving the diffusion part explicitly causes the stability condition for the time-step to depend on the spatial discretization.

Second Order Semi-Implicit Backward Differentiation

This multistep method usually denoted SBDF is a semi-implicit version of the backward differentiation formula (BDF) and is also known as extrapolated Gear. It has been studied in the case of time-dependent PDEs in [1] and for the bidomain model in [8]. As for the Forward-Backward Euler's method, the terms are taken explicitly other than the diffusion term.

The Rush Larsen Method

A very popular method for solving physiological models is the first-order scheme pro-

posed by Rush and Larsen [28]. For simplicity we will introduce the method for the following scalar initial value problem,

$$\begin{cases} \dfrac{dy}{dt} = f(t,y) = a(t,y)y + b(t,y), \quad t \in (0,T], \\ y(0) = y^0. \end{cases} \tag{2.3.15}$$

In general, there are different ways of choosing a and b, depending on the ionic model used. The idea behind the method is to recognize that by assuming that a and b are constant, the solution of (2.3.15) after a time increment of Δt is a simple exponential of the form:

$$y(\Delta t) = e^{a\Delta t}\left(y^0 + \frac{b}{a}\right) - \frac{b}{a}. \tag{2.3.16}$$

We then utilize (2.3.16) to construct an iterative scheme by taking the obtained solution as the initial condition for the next time-step, updating the values of a and b at each iteration. The proposed numerical scheme for the solution of (2.3.15) is given by

$$\begin{cases} y^{n+1} = e^{a^n\Delta t}\left(y^n + \dfrac{b^n}{a^n}\right) - \dfrac{b^n}{a^n}, \quad n = 0, ...N_t, \\ y(0) = y^0, \end{cases} \tag{2.3.17}$$

where a^n and b^n are defined as $a^n = a(t_n, y^n)$ and $b^n = b(t_n, y^n)$, respectively.
This original Rush-Larsen scheme is only first-order accurate. Fortunately, Perego and Veneziani [24] introduced a way to increase the accuracy of the scheme, by taking a and b at time $t_{n+\frac{1}{2}}$,

$$\begin{cases} y^{n+1} = y^n + \Delta t\Phi(a^{n+\frac{1}{2}}\Delta t)(a^{n+\frac{1}{2}}y^n + b^{n+\frac{1}{2}}), \quad n = 0, ...N_t, \\ y(0) = y^0, \end{cases} \tag{2.3.18}$$

where

$$\Phi(x) = \begin{cases} \dfrac{e^x - 1}{x}, & x \neq 0, \\ 1, & x = 0, \end{cases}$$

$a^{n+\frac{1}{2}}$ and $b^{n+\frac{1}{2}}$ are approximations of $a(t_{n+\frac{1}{2}})$ and $b(t_{n+\frac{1}{2}})$ taken as

$$\begin{aligned} &a^{n+\frac{1}{2}} = c_{-1}a^{n+1} + c_0 a^n + c_1 a^{n-1}, \quad b^{n+\frac{1}{2}} = c_{-1}b^{n+1} + c_0 b^n + c_1 b^{n-1},\ n = 1, ..., N_t, \\ &a^{\frac{1}{2}} = c_{-1}a^1 + (c_0 + c_1)a^0, \quad b^{\frac{1}{2}} = c_{-1}b^1 + (c_0 + c_1)b^0, \end{aligned} \tag{2.3.19}$$

where c_{-1}, c_0 and c_1 are coefficients to be determined.

For the problem (1.2.6)-(1.2.8), we want to apply the Rush-Larsen Method only on the gating variables. We need to write the function $\mathbf{f}$ in (1.2.7) as $f_i(u, \mathbf{v}, \mathbf{X}) = a_i(u)v_i + b_i(u)$. It is necessary that the ionic model used has f_i depending only on u and v_i, which is the case for most models. For the BR and TNNP models, we have:

$$a_i(u) = \frac{-1}{\tau_{v_i}(u)} \quad \text{and} \quad b_i(u) = \frac{v_{i,\infty}(u)}{\tau_{v_i}(u)}.$$

The concentration ODEs are solved using an explicit method: Forward Euler for the first-order method, and Adams-Bashforth 2 for the second-order method. The transmembrane PDE is solved using semi-implicit methods: Forward-Backward Euler for the first-order method and Crank-Nicolson with Adams-Bashforth 2 for the second-order method.

Operator Splitting

Operator splitting is a useful technique when solving coupled systems of PDEs. It involves splitting complex equation systems into smaller parts that are easier to solve, for example splitting the reaction and diffusion parts in a reaction-diffusion equation. For simplicity we will introduce our method for the following initial value problem,

$$\begin{cases} \dfrac{dy}{dt} = (L_1 + L_2)y, \\ y(0) = y_0, \end{cases} \tag{2.3.20}$$

where L_1 and L_2 are operators acting on y. Choosing a small time-step Δt, an approximate solution of the problem at time $t = \Delta t$ can be computed in two steps:

Step 1: Solving the problem

$$\begin{cases} \dfrac{du}{dt} = L_1 u, \\ u(0) = y_0, \end{cases} \tag{2.3.21}$$

for $t \in [0, \Delta t]$.

Step 2: Solving the problem

$$\begin{cases} \dfrac{dv}{dt} = L_2 v, \\ v(0) = u(\Delta t), \end{cases} \tag{2.3.22}$$

for $t \in [0, \Delta t]$ using as initial condition for v the solution obtained in the first step.The approximation of $y(\Delta t)$ is the solution from step two, $v(\Delta t)$.

This method is called Godunov splitting and is first-order accurate in the time-step Δt [9, 34]. We can modify this method to obtain a second-order accurate method consisting of these three steps:

Step 1: Solving the problem

$$\begin{cases} \dfrac{du}{dt} = L_1 u, \\ u(0) = y^0, \end{cases} \tag{2.3.23}$$

for $t \in [0, \Delta t/2]$.

Step 2: Solving the problem

$$\begin{cases} \dfrac{dv}{dt} = L_2 v, \\ v(0) = u(\Delta t/2), \end{cases} \tag{2.3.24}$$

for $t \in [0, \Delta t]$.

Step 3: Solving the problem

$$\begin{cases} \dfrac{du}{dt} = L_1 u, \\ u(\Delta/2) = v(\Delta t), \end{cases} \tag{2.3.25}$$

for $t \in [\Delta t/2, \Delta t]$. The approximation of $y(\Delta t)$ is the solution from step three, $u(\Delta t)$. This method is called Strang splitting [32, 34]. For the studied problem, L_2 represents the discretized diffusion term and L_1, the functions from the ionic model. For

our Strang splitting scheme, we use explicit Runge-Kutta (RK) methods for Steps 1 and 3, and the implicit Crank-Nicolson (CN) method for Step 2. This results in a semi-implicit method.

Remark. To reduce the computational cost of this Strang splitting method, Step 3 can be combined with the Step 1 of the next time-step. Instead of computing two half steps of RK, one could compute one RK step. To preserve the second order, the last time-step would have to end with a half step of RK. This would reduce the number of times the ionic functions are computed but it would also change the stability condition.

Deferred Correction

Deferred (or defect) correction methods [19] can be constructed for any order k. Considering the problem (2.1.1), we want to construct an approximation of y at time t_n by summing successive corrections. The approximation of order k for this method is constructed as $y^n = y_0^n + \Delta t y_1^n + \Delta t^2 y_2^n + ... + \Delta t^k y_k^n + \mathcal{O}(\Delta t^{k+1})$, where the corrections $y_0^n, ..., y_k^n$ are computed one after the other. In our case we will use Forward-Backward Euler steps for the corrections. This method has been used in the case of Navier-Stokes equations in [10] and has not yet been tested for the monodomain or bidomain models. The method proposed above is a semi-implicit third-order deferred correction scheme for the monodomain model. Using only the first two correction steps written above is possible and gives a second-order approximation of the solution.

First Iteration for Multi-Step Methods

Note that for multi-step methods, we need to initiate the first iteration using a one-step method. In fact, these methods require the value of the solution at the two previous time-steps, which are not available at time $t = t_1 = \Delta t$. We only have the initial value of the solution at time t_0. A solution to this problem is to take $Y^{-1} = Y^0$ for the first iteration, which changes the multi-step method into a one-step method.

However, this does not generally ensure that the method remains consistent. For the considered schemes, we have such problems for the SBDF2 method. For the first step, this strategy gives the following scheme which is non-consistent:

$$\begin{aligned}\frac{\frac{3}{2}U^1-\frac{3}{2}U^0}{\Delta t} &= -I(U^0,V^0,X^0)+AU^1,\\ \frac{\frac{3}{2}V^1-\frac{3}{2}V^0}{\Delta t} &= F(U^0,V^n,X^0),\\ \frac{\frac{3}{2}X^1-\frac{3}{2}X^0}{\Delta t} &= G(U^0,V^0,X^0).\end{aligned}$$

Instead, we need to use a one-step method, such as FBE, for the first time-step. As for the RL-CNAB method, by taking $Y^{-1}=Y^0$, we end up with the RL-FBE method. In some programs, it might not be simple to change the time-stepping method for the first iteration of the method. This was the case in the Fortran code that we used to make 2D simulations. When using the finite element method, the first equation of (2.3.4) for the SBDF2 method would then be written as:

$$\left(\frac{3}{2}M-\Delta tA\right)U^{n+1}=M\left(2U^n-\frac{1}{2}U^{n-1}+\Delta t(-2I(Y^n)+I(Y^{n-1}))\right), \quad (2.3.26)$$

where M is the mass matrix and A is the stiffness matrix. The left-hand-size matrix multiplying U^{n+1} is computed at the beginning of the program. It is not simple to replace the first iteration of SBDF2 by FBE because the first equation of (2.3.2) is written as

$$(M-\Delta tA)\,U^{n+1}=M\,(U^n+\Delta tI(Y^n))\,. \quad (2.3.27)$$

We see that the left-hand-side matrix is different, and then one must find a one-step method using the same left-hand-side matrix. To do so, we used the following θ-scheme:

$$\left(\frac{3}{2}M-\Delta tA\right)U^{n+1}=M\left(\frac{3}{2}U^n+\Delta tI(Y^n)\right)+\frac{\Delta t}{2}AU^n, \quad (2.3.28)$$

which is consistent but does not require a recalculation of the left-hand-side matrix. Furthermore, the choice of the method used for the first iteration is important when

considering the order of convergence of the method. We know that first-order methods are locally of second-order, and that second-order methods are locally of third-order. Therefore, if we take a first-order method for the first iteration of our second-order time-stepping scheme, we conserve the order of convergence. The same idea applies to higher order methods: third-order methods only require a second-order scheme for the first time-step. Note that the DC3 method is self-starting, so it does not require another method for the first time-step.

Chapter 3

Stability Analysis

In this chapter, we find stability conditions for the numerical methods introduced in the last chapter.

3.1 Energy method

In this section, the stability analysis will be done in a PDE setting, where we use the energy method to find an upper bound on the time-step Δt needed for stability. Though the derivation is very similar to what was done in [8] for the FitzHugh-Nagumo model coupled with the bidomain model, we present the derivation of stability estimates with the energy method to explain the stability conditions of numerical methods in the case of a general ionic model coupled with the monodomain model. We wish to demonstrate that our numerical solutions are stable, i.e. they remain bounded for sufficiently small values of the time-step Δt, and to determine the conditions on this time-step. We consider the stability of the problem (1.2.9)-(1.2.10).

Lipschitz Condition

We suppose that I_{ion} and f_i satisfy the following Lipschitz condition

$$\|I_{ion}(u, \mathbf{v})\| \leq L_I[\|u\|_0 + \|\mathbf{v}\|_0],$$

$$\|f_i(u,\mathbf{v})\| \leq L_{f_i}[\|u\|_0 + \|\mathbf{v}\|_0] \quad \text{for } i = 1,\ldots,p+q,$$

where $L_I, L_{f_i} \geq 0$ and $\|\mathbf{v}\|_0 = \left(\sum_{j=1}^{p+q} \|v_j\|_0^2\right)^{1/2}$. Functions I_{ion} and f_i used in some ionic models are locally Lipschitz and as solutions remain bounded practically (we mean in computations), we can identify constants L_I and L_{f_i} for these estimates to provide useful information on stability.

We consider the the variational formulation of the problem: Find $(u,v) \in C^1(0,T;V_h^{p+q+1})$ such that

$$\int_\Omega u_t \Phi^h dx = -\int_\Omega I_{ion}(u,\mathbf{v})\Phi^h dx - \int_\Omega \sigma \nabla u \nabla \Phi^h dx, \tag{3.1.1}$$

$$\int_\Omega v_{i,t}\varphi_i^h dx = \int_\Omega f_i(u,\mathbf{v})\varphi_i^h dx, \quad \text{for } i = 1,\ldots,p+q, \tag{3.1.2}$$

for all $\Phi^h, \varphi_1^h, \ldots \varphi_{p+q}^h \in V_h$, where V_h is the space of P_1 linear continuous finite elements.

3.1.1 Forward Euler

We apply the FE time-stepping method to the problem (3.1.1)-(3.1.2) and take $\Phi^h = u^{n+1}$ and $\phi_i^h = v_i^{n+1}$ for each i as test functions. We obtain

$$\int_\Omega \left(\frac{u^{n+1}-u^n}{\Delta t} + I_{ion}(u^n,\mathbf{v}^n)\right) u^{n+1} dx + \int_\Omega \sigma \nabla u^n \nabla u^{n+1} dx = 0,$$

$$\int_\Omega \left(\frac{\mathbf{v}^{n+1}-\mathbf{v}^n}{\Delta t} - f(u^n,\mathbf{v}^n)\right)\cdot \mathbf{v}^{n+1} dx = 0.$$

We can bound below the diffusive term

$$\begin{aligned}\int_\Omega \sigma \nabla u^n \nabla u^{n+1} dx &= \sigma\left(\int_\Omega \nabla^2 u^{n+1} dx - \int_\Omega \left(\nabla u^{n+1} - \nabla u^n\right)\nabla u^{n+1} dx\right)\\ &\geq \sigma\left(|u^{n+1}|_1^2 - |u^{n+1}-u^n|_1 |u^{n+1}|_1\right),\end{aligned}$$

where $|.|_1$ is the H^1 semi-norm. Using the identity

$$2(a^{n+1}-a^n)a^{n+1} = (a^{n+1})^2 + (a^{n+1}-a^n)^2 - (a^n)^2,$$

we write

$$\begin{aligned}
&\|u^{n+1}\|_0^2+\|u^{n+1}-u^n\|_0^2-\|u^n\|_0^2+\|\mathbf{v}^{n+1}\|_0^2+\|\mathbf{v}^{n+1}-\mathbf{v}^n\|_0^2-\|\mathbf{v}^n\|_0^2\\
&\qquad+2\Delta t\sigma|u^{n+1}|_1^2-2\Delta t\sigma|u^{n+1}-u^n|_1|u^{n+1}|_1\\
&\qquad\le 2\int_\Omega\left(u^{n+1}-u^n\right)u^{n+1}dx+2\Delta t\int_\Omega\sigma\nabla u^n\nabla u^{n+1}dx+2\sum_{i=1}^{p+q}\int_\Omega\left(v_i^{n+1}-v_i^n\right)v_i^{n+1}dx\\
&\qquad=-2\Delta t\int_\Omega I_{ion}(u^n,\mathbf{v}^n)u^{n+1}dx+2\Delta t\sum_{i=1}^{p+q}\int_\Omega f_i(u^n,\mathbf{v}^n)v_i^{n+1}dx
\end{aligned}$$

From an inverse inequality in [6, Th. 3.2.6. on p.140] we get:

$$|u^{n+1}-u^n|_1\le\frac{\mathscr{C}}{h}\|u^{n+1}-u^n\|_0. \tag{3.1.3}$$

Using our Lipschitz condition, Hölder's inequality and Young's inequality, we have

$$\begin{aligned}
-\int_\Omega I_{ion}(u^n,\mathbf{v}^n)u^{n+1}dx&\le L_I[\|u^n\|_0+\|\mathbf{v}^n\|_0]\|u^{n+1}\|_0\\
&\le L_I\left[\frac{1}{2}\|u^n\|_0^2+\|u^{n+1}\|_0^2+\frac{1}{2}\|\mathbf{v}^n\|_0^2\right].
\end{aligned}$$

A similar result is obtained for the terms in f_i. Using Young's inequality and the inverse inequality (3.1.3) , we have

$$\begin{aligned}
2\Delta t\sigma[|u^{n+1}|_1^2-|u^{n+1}-u^n|_1|u^{n+1}|_1]&\ge 2\Delta t\left[\sigma|u^{n+1}|_1^2-\frac{s}{2}|u^{n+1}|_1^2-\frac{\sigma^2}{2s}|u^{n+1}-u^n|_1^2\right]\\
&\ge\Delta t(2\sigma-s)|u^{n+1}|_1^2-\Delta t\frac{\sigma^2}{s}\frac{\mathscr{C}^2}{h^2}\|u^{n+1}-u^n\|_0^2.
\end{aligned}$$

Putting all this together, we obtain:

$$\begin{aligned}
&\|u^{n+1}\|_0^2+\left(1-\Delta t\frac{\mathscr{C}^2\sigma^2}{sh^2}\right)\|u^{n+1}-u^n\|_0^2+\|\mathbf{v}^{n+1}\|_0^2+\Delta t(2\sigma-s)|u^{n+1}|_1^2\le\\
&\|u^n\|_0^2+\|\mathbf{v}^n\|_0^2+\Delta tC_1\|u^{n+1}\|_0^2+\Delta tC_2\|\mathbf{v}^{n+1}\|+\Delta tC_3\|u^n\|_0^2+\Delta tC_4\|\mathbf{v}^n\|_0^2,
\end{aligned}\tag{3.1.4}$$

where $C_1=2L_I$, $C_2=2\left(\max\limits_{i=1,\dots,p+q}L_{f_i}\right)$ and $C_3=C_4=L_I+\sum_{i=1}^{p+q}L_{f_i}$. To make sure all terms on the left-hand side of the inequality are positive, we restrict s and Δt so that $s<2\sigma$ and $1-\Delta t\dfrac{\mathscr{C}^2\sigma^2}{sh^2}\ge 0$, i.e.

$$\Delta t\le\frac{h^2s}{\mathscr{C}^2\sigma^2}<\frac{2h^2}{\mathscr{C}^2\sigma}. \tag{3.1.5}$$

We sum the inequality (3.1.4) for n going from 0 to $m-1$, where m is an integer between 1 and N_t. Let $k = 2\sigma - s$, $C = \max\{C_1, C_2\}$, and $K = \max\{C_1+C_3, C_2+C_4\}$. We have the following result:

$$(1-\Delta tC)[\|u^m\|_0^2 + \|\mathbf{v}^m\|_0^2] + \Delta t\ k\sum_{n=1}^{m}|u^n|_1^2 \leq \|u^0\|_0^2 + \|\mathbf{v}^0\|_0^2 + \Delta tK\sum_{n=0}^{m-1}(\|u^n\|_0^2 + \|\mathbf{v}^n\|_0^2) \tag{3.1.6}$$

We choose $\Delta t < 1/C$ and apply the discrete Gronwall lemma to (3.1.6) to obtain

$$\max_{n=1,\dots N_t}\{\|u^n\|_0^2 + \|\mathbf{v}^2\|_0^2\} \leq \frac{1}{1-\Delta tC}(\|u^0\|_0^2 + \|\mathbf{v}^0\|_0^2)e^{TK/(1-\Delta tC)}. \tag{3.1.7}$$

By combining (3.1.6) and (3.1.7), we write

$$\sum_{n=1}^{N_t}\Delta t|u^n|_1^2 \leq \frac{1}{k}(\|u^0\|_0^2 + \|\mathbf{v}^0\|_0^2) + \frac{TK}{k}\max_n\{\|u^n\|_0^2 + \|\mathbf{v}^n\|_0^2\}.$$

Let us extend the numerical solutions at discrete time t_n to functions u and $\mathbf{v}$ that are piecewise constant in time such that $u(n\Delta t) = u^n$, $\mathbf{v}(n\Delta t) = \mathbf{v}^n$ for $n = 0, \dots, N_t$. If the conditions imposed on Δt are met, u and $\mathbf{v}$ remain stable in the spaces $L^\infty(0,T;L^2(\Omega)) \cap L^2(0,T;H^1(\Omega))$ and $L^\infty(0,T;L^2(\Omega))$, respectively. These conditions are $\Delta t < 1/C$, with $C = O(\max\{L_I, L_f\})$ and condition (3.1.5), where $L_f = \max_{i=1,\dots,p+q} L_{f_i}$. Condition (3.1.5) is similar to the usual stability condition for FE applied to the heat equation.

Discussions on Forward Euler method's stability conditions

An estimate was found for the constant $\mathscr{C}$ in [8]. In the 1D case, we get $\mathscr{C} \geq 2\sqrt{3}$. The condition (3.1.5) becomes:

$$\Delta t < \frac{h^2}{6\sigma}.$$

3.1.2 Second Order Semi-Implicit Backward Differentiation

We apply the SBDF2 time-stepping method to the problem (3.1.1)-(3.1.2) and take $\Phi^h = u^{n+1}$ and $\phi_i^h = v_i^{n+1}$ for each i as test functions. We obtain

$$\int_\Omega \left(\frac{3u^{n+1} - 4u^n + u^{n-1}}{2\Delta t} + (2I_{ion}(u^n, \mathbf{v}^n) - I_{ion}(u^{n-1}, \mathbf{v}^{n-1})) \right) u^{n+1} dx + \int_\Omega \sigma \nabla^2 u^{n+1} dx = 0,$$

$$\int_\Omega \left(\frac{3v_i^{n+1} - 4v_i^n + v_i^{n-1}}{2\Delta t} - (2f_i(u^n, \mathbf{v}^n) - f_i(u^{n-1}, \mathbf{v}^{n-1})) \right) v_i^{n+1} dx = 0 \quad \text{for } i = 1, \ldots, p+q.$$

Using the identity

$$\begin{aligned}(6a^{n+1} - 8a^n + 2a^{n-1})a^{n+1} =& (a^{n+1})^2 + (2a^{n+1} - a^n)^2 + (\delta_{tt} a^{n+1})^2 \\ & - (a^n)^2 - (2a^n - a^{n-1})^2,\end{aligned}$$

where $\delta_{tt} a^{n+1} = a^{n+1} - 2a^n + a^{n-1}$, we obtain

$$\begin{aligned}\|u^{n+1}\|_0^2 + & \|2u^{n+1} - u^n\|_0^2 - \|u^n\|_0^2 - \|2u^n - u^{n-1}\|_0^2 + \|\mathbf{v}^{n+1}\|_0^2 \\ & + \|2\mathbf{v}^{n+1} - u^n\|_0^2 - \|\mathbf{v}^n\|_0^2 + 4\Delta t\sigma |u^{n+1}|_1^2 \\ \leq & \Delta t C_1 \|u^{n+1}\|_0^2 + \Delta t C_2 \|\mathbf{v}^{n+1}\|_0^2 + \Delta t C_3 \|u^n\|_0^2 + \Delta t C_4 \|\mathbf{v}^n\|_0^2 \\ & + \Delta t C_5 \|u^{n-1}\|_0^2 + \Delta t C_6 \|\mathbf{v}^{n-1}\|_0^2,\end{aligned}$$

where $C_1 = 12L_I$, $C_2 = 12L_f$, $C_3 = C_4 = 4(L_I + \sum_{i=1}^{p+q} L_{f_i})$ and $C_5 = C_6 = 2(L_I + \sum_{i=1}^{p+q} L_{f_i})$. Summing the inequality for n from 0 to $m-1$, we get u^{-1} and $\mathbf{v}^{-1}$ values that we will take equal to u^0 and $\mathbf{v}^0$, respectively. As we have discussed in the previous chapter, this does not result in a consistent method for the first iteration. However, this makes the stability argument simpler but the conclusion remains valid for the consistent method. Taking $C = \max\{C_1, C_2\}$ and $K = \max\{C_1 + C_3 + 2C_5, C_2 + C_4 + 2C_6\}$, we get:

$$\begin{aligned}(1 - \Delta t C)[\|u^m\|_0^2 + \|\mathbf{v}^m\|_0^2] + 4\Delta t\sigma \sum_{n=1}^{m} |u^n|_1^2 \\ \leq 2(\|u^0\|_0^2 + \|\mathbf{v}^0\|_0^2) + \Delta t K \sum_{n=0}^{m-1} (\|u^n\|_0^2 + \|\mathbf{v}^n\|_0^2).\end{aligned} \tag{3.1.8}$$

Choosing Δt such that $\Delta t < 1/C$ and applying the discrete Gronwall Lemma to (3.1.8) we get:

$$\max_{n=1,\ldots,N_t} \{\|u^n\|_0^2 + \|\mathbf{v}^n\|_0^2\} \leq \frac{2}{1 - \Delta t C} [\|u^0\|_0^2 + \|\mathbf{v}^0\|_0^2] e^{TK/(1-\Delta t C)}. \tag{3.1.9}$$

By combining (3.1.8) and (3.1.9) we have

$$\sum_{n=1}^{N_t} \Delta t |u^n|_1^2 \leq \frac{1}{2\sigma}(\|u^0\|_0^2 + \|\mathbf{v}^0\|_0^2) + \frac{TK}{4\sigma} \max_n \{\|u^n\|_0^2 + \|\mathbf{v}^n\|_0^2\} \tag{3.1.10}$$

Therefore the discrete functions u and $\mathbf{v}$ remain stable in the spaces $L^\infty(0,T;L^2(\Omega)) \cap L^2(0,T;H^1(\Omega))$ and $L^\infty(0,T;L^2(\Omega))$, respectively, with the condition that $\Delta t < 1/C$ holds. C depends on C_1 and C_2, with $C_1 = O(L_I)$ and $C_2 = O(L_f)$, then $C = O(\max\{L_I, L_f\})$

For many ionic models that are locally Lipschitz, Lipschitz constants valid on the bounded region holding the solution are not easily identified and we must find an upper bound for them. As we will see in Section 3.3, the given condition on Δt is too restrictive and far from the actual numerically observed condition on stability. In the case of the FitzHugh-Nagumo model, we can find a better bound than the one derived from the Lipschitz condition and thus obtain a better bound on Δt [8].

3.2 Absolute stability

In this section, the stability analysis will be done using the concept of absolute stability for ODE solvers. We will linearize our PDE and then consider the linear ODE system given by the linearized problem semi-discretized in space. We will find the stability region of each method for the linearized problem. The largest possible time-step Δt for which $\lambda \Delta t$ is in the stability region is called the critical time-step.

3.2.1 Forward Euler

Consider the general ionic model coupled with the monodomain model that we have nondimensionalized in Chapter 1 in equations (1.2.9)-(1.2.10):

$$\frac{\partial u}{\partial t} = -I_{ion}(u, \mathbf{v}) + \sigma \Delta u, \tag{3.2.1}$$

$$\frac{\partial \mathbf{v}}{\partial t} = \mathbf{f}(u, \mathbf{v}), \tag{3.2.2}$$

We linearize around some constant state $(\tilde{u}, \tilde{\mathbf{v}})$ that will be determined later. Equations (3.2.1) and (3.2.2) become

$$\frac{\partial u}{\partial t} = -\frac{\partial I_{ion}}{\partial u}(\tilde{u}, \tilde{\mathbf{v}})u - \frac{\partial I_{ion}}{\partial v}(\tilde{u}, \tilde{\mathbf{v}})v + \sigma \Delta u, \tag{3.2.3}$$

$$\frac{\partial \mathbf{v}}{\partial t} = \frac{\partial \mathbf{f}}{\partial u}(\tilde{u}, \tilde{\mathbf{v}})u + \frac{\partial \mathbf{f}}{\partial v}(\tilde{u}, \tilde{\mathbf{v}})v, \tag{3.2.4}$$

For the sake of the stability analysis, we consider evenly spaced grid points $\ldots, x_{-1}, x_0, x_1, \ldots$ covering the real line with $x_j = jh$ for each $j \in \mathbb{Z}$. We denote $U = [\ldots, U_{-1}, U_0, U_1, \ldots]^T \simeq [\ldots, u(x_{-1}, t), u(x_0, t), u(x_1, t), \ldots]^T$ and $V = [\ldots, V_{-1}, V_0, V_1, \ldots]^T \simeq [\ldots, v(x_{-1}, t), v(x_0, t), v(x_1, t), \ldots]^T$, where $v(x, t) = [v_1(x, t), \ldots, v_{p+q}(x, t)]^T$. Putting (3.2.3) and (3.2.4) together we obtain

$$\frac{d}{dt}\begin{bmatrix} U_j \\ V_j \end{bmatrix} = \begin{bmatrix} -\frac{\partial I_{ion}}{\partial u}(\tilde{u}, \tilde{\mathbf{v}}) & -\frac{\partial I_{ion}}{\partial \mathbf{v}}(\tilde{u}, \tilde{\mathbf{v}}) \\ \frac{\partial \mathbf{f}}{\partial u}(\tilde{u}, \tilde{\mathbf{v}}) & \frac{\partial \mathbf{f}}{\partial v}(\tilde{u}, \tilde{\mathbf{v}}) \end{bmatrix} \begin{bmatrix} U_j \\ V_j \end{bmatrix} + \sigma \begin{bmatrix} \frac{U_{j+1} - 2U_j + U_{j-1}}{h^2} \\ 0 \end{bmatrix}, \tag{3.2.5}$$

for each $j \in \mathbb{Z}$. We set $U_j(t) = U_\omega(t)e^{i\omega jh}$ and $V_j(t) = V_\omega(t)e^{i\omega jh}$, for all $\omega \in [0, \frac{2\pi}{h}]$, as is usually done for von Neumann stability analysis. Then

$$\frac{d}{dt}\begin{bmatrix} U_\omega \\ V_\omega \end{bmatrix} = J \begin{bmatrix} U_\omega \\ V_\omega \end{bmatrix} + \sigma \begin{bmatrix} \frac{e^{i\omega h} - 2 + e^{-i\omega h}}{h^2} U_\omega \\ 0, \end{bmatrix} \tag{3.2.6}$$

where J is the Jacobian matrix evaluated at $(\tilde{u}, \tilde{\mathbf{v}})$ as in (3.2.5). If we now discretize in time and apply the FE scheme on (3.2.6), it becomes

$$\left(\begin{bmatrix} U_\omega^{n+1} \\ V_\omega^{n+1} \end{bmatrix} - \begin{bmatrix} U_\omega^{n} \\ V_\omega^{n} \end{bmatrix} \right) \frac{1}{\Delta t} = J \begin{bmatrix} U_\omega^{n} \\ V_\omega^{n} \end{bmatrix} + \sigma \begin{bmatrix} \frac{2\cos \omega h - 2}{h^2} U_\omega^n \\ 0 \end{bmatrix}, \tag{3.2.7}$$

$\Longleftrightarrow$

$$\begin{bmatrix} U_\omega^{n+1} \\ V_\omega^{n+1} \end{bmatrix} = \begin{bmatrix} U_\omega^{n} \\ V_\omega^{n} \end{bmatrix} + \Delta t \left(J \begin{bmatrix} U_\omega^{n} \\ V_\omega^{n} \end{bmatrix} + \begin{bmatrix} \sigma \frac{2\cos \omega h - 2}{h^2} & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} U_\omega^{n} \\ V_\omega^{n} \end{bmatrix} \right). \tag{3.2.8}$$

Denoting A_ω the diffusion matrix occuring in the last term on the right-hand-side,

$$\begin{bmatrix} U_\omega^{n+1} \\ V_\omega^{n+1} \end{bmatrix} = (I + \Delta t(J + A_\omega)) \begin{bmatrix} U_\omega^{n} \\ V_\omega^{n} \end{bmatrix}. \tag{3.2.9}$$

We define the stability function $R(\Delta t, h, \omega) = I + \Delta t(J + A_\omega)$. For stability we need that the spectral radius $\rho(R(\Delta t, h, \omega)) \leq 1$. This is implied by having $\rho(\Delta t(J+A_\omega)) \leq 2$. Indeed, looking at the eigenvalues of the Jacobian for the different ionic models [22] and the eigenvalues of the diffusion matrix, we see that the spectral radius depends on the most negative eigenvalue of $J+A_\omega$. The condition for stability is then $\Delta t \leq -2/\lambda_{min}$, where λ_{min} is the most negative eigenvalue of $J + A_\omega$. Because this eigenvalue depends on ω, we will consider the wave number ω that makes this inequality most restrictive, i.e. $\omega = \pi/h$, a fact that can be verified numerically. Therefore our critical time-step for the stability of the FE method is $\Delta t^*_{theo} = -2/\lambda_{min}$, where λ_{min} is the minimum eigenvalue of

$$\begin{bmatrix} -\dfrac{\partial I_{ion}}{\partial u}(\tilde{u}, \tilde{\mathbf{v}}) - \dfrac{4\sigma}{h^2} & -\dfrac{\partial I_{ion}}{\partial \mathbf{v}}(\tilde{u}, \tilde{\mathbf{v}}) \\ \dfrac{\partial \mathbf{f}}{\partial u}(\tilde{u}, \tilde{\mathbf{v}}) & \dfrac{\partial \mathbf{f}}{\partial \mathbf{v}}(\tilde{u}, \tilde{\mathbf{v}}) \end{bmatrix}. \tag{3.2.10}$$

To determine λ_{min}, we calculate numerically the solution of the problem on a fine grid. We then evaluate the matrix $J+A_\omega$ at each node of our domain, and for each of these matrices, we calculate their eigenvalues. We choose the most negative eigenvalue as our λ_{min}. The constant state $(\tilde{u}, \tilde{\mathbf{v}})$ is then chosen as the numerical solution evaluated at this node. The most negative eigenvalue typically appears during the resting state of the solution, but for some models, the stimulation current can sometimes make it more negative. The theoretical critical time-steps are shown in Table 3.2 for the BR model, Table 3.5 for the MS model and in Table 3.8 for the TNNP model. We can see that with small values of h, the critical time-step is proportional to h^2.

Numerical verification for the choice of ω

We can verify numerically that for different choices of $\omega \in [0, \frac{2\pi}{h}]$, $\rho(R(\Delta t^*_{theo}, h, \omega)) \leq 1$ for J evaluated at any node of the discretized domain. Therefore the stability condition $\Delta t \leq -2/\lambda_{min}$ is sufficient, at least from numerical verifications.

3.2.2 Forward-Backward Euler

Following the same approach, we write the FBE scheme as:

$$\left(\begin{bmatrix} U_\omega^{n+1} \\ V_\omega^{n+1} \end{bmatrix} - \begin{bmatrix} U_\omega^{n} \\ V_\omega^{n} \end{bmatrix} \right) \frac{1}{\Delta t} = J \begin{bmatrix} U_\omega^{n} \\ V_\omega^{n} \end{bmatrix} + A_\omega \begin{bmatrix} U_\omega^{n+1} \\ V_\omega^{n+1} \end{bmatrix} \tag{3.2.11}$$

$\iff$

$$(I - \Delta t A_\omega) \begin{bmatrix} U_\omega^{n+1} \\ V_\omega^{n+1} \end{bmatrix} = (I + \Delta t J) \begin{bmatrix} U_\omega^{n} \\ V_\omega^{n} \end{bmatrix} \tag{3.2.12}$$

$\iff$

$$\begin{bmatrix} U_\omega^{n+1} \\ V_\omega^{n+1} \end{bmatrix} = (I - \Delta t A_\omega)^{-1}(I + \Delta t J) \begin{bmatrix} U_\omega^{n} \\ V_\omega^{n} \end{bmatrix} \tag{3.2.13}$$

We have the stability function $R(\Delta t, h, \omega) = (I - \Delta t A_\omega)^{-1}(I + \Delta t J)$. For stability we need $\rho(R(\Delta t, h, \omega)) \leq 1$. This stability condition depends on $\omega \in [0, \frac{2\pi}{h}]$ and therefore we look for the value of ω which is more restrictive for the stability. One can easily see that the matrix $(I - \Delta t A_\omega)^{-1}$ is diagonal with value $\left(1 - \Delta t \sigma \frac{2\cos\omega h - 2}{h^2}\right)^{-1}$ in the first position and 1 on the rest of the diagonal. This value is always in the interval $(0, 1]$. The most restrictive case for stability is when this value is 1, i.e. when $\omega = 0$. This can be verified numerically. The condition for stability then becomes $\rho(I + \Delta t J) \leq 1$. Therefore, similarly to the FE method, our critical time-step for FBE is $\Delta t^*_{theo} = -2/\lambda_{min}$, where λ_{min} is the most negative eigenvalue of J. As explained for the previous method, we calculate J at each node of a fine grid and we choose its most negative eigenvalue as our λ_{min}. The constant state $(\tilde{u}, \tilde{\mathbf{v}})$ is then chosen as the numerical solution evaluated at that node. The critical time-steps for the BR, MS and TNNP models are shown in Tables 3.2, 3.5 and 3.7, respectively. On coarse

meshes, FE and FBE are expected to be stable for the same time-step Δt, but as the grid is refined, FE's stability deteriorates while FBE's remains unchanged. In fact, the diffusion term taken implicitly in the FBE method makes the critical time-step independent from the grid size h.

Remark. We can look at the problem (3.2.1)-(3.2.2) by linearizing it after the spatial discretization. Discretizing in space on a finite interval and linearizing we have:

$$\frac{\partial}{\partial t}\begin{bmatrix} u_1 \\ v_1^1 \\ v_1^2 \\ \vdots \\ v_1^p \\ u_2 \\ \vdots \\ v_N^p \end{bmatrix} = \begin{bmatrix} J_0 & 0 & \cdots & 0 \\ 0 & J_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & J_N \end{bmatrix} \begin{bmatrix} u_1 \\ v_1^1 \\ v_1^2 \\ \vdots \\ v_1^p \\ u_2 \\ \vdots \\ v_N^p \end{bmatrix} + A \begin{bmatrix} u_1 \\ v_1^1 \\ v_1^2 \\ \vdots \\ v_1^p \\ u_2 \\ \vdots \\ v_N^p \end{bmatrix}, \tag{3.2.14}$$

where A is a discrete diffusion matrix and for each $i = 0, \dots N$, J_i is the Jacobian matrix at the node x_i, i.e.

$$J_i = \begin{bmatrix} -\dfrac{\partial I_{ion}}{\partial u}(u_i, \mathbf{v}_i) & -\dfrac{\partial I_{ion}}{\partial \mathbf{v}}(u_i, \mathbf{v}_i) \\ \dfrac{\partial \mathbf{f}}{\partial u}(u_i, \mathbf{v}_i) & \dfrac{\partial \mathbf{f}}{\partial \mathbf{v}}(u_i, \mathbf{v}_i) \end{bmatrix}.$$

Denoting Y as the solution vector, then A is such that the components of AY are $\sigma \dfrac{u_{j+1} - 2u_j + u_{j-1}}{h^2}$ for $j = 1, \dots, N-1$, at the position corresponding to $\dfrac{\partial u_j}{\partial t}$ and 0 otherwise.

Applying the Forward-Backward Euler scheme to (3.2.14), we obtain

$$Y^{n+1} = (I - \Delta t A)^{-1}(I + \Delta t J)Y^n, \tag{3.2.15}$$

where Y^n and Y^{n+1} are the solution vectors at time t_n and t_{n+1} and J is the block diagonal matrix of the Jacobians. For stability, we need $\rho(R(\Delta t, J)) \leq 1$, where

$R(\Delta t, J) = (I - \Delta t A)^{-1}(I + \Delta t J)$. We will only consider a sufficient condition for l^∞ stability, i.e. $\|R(\Delta t, J)\| \leq 1$. This condition is sufficient but not necessary as for all induced norms, $\rho(R) \leq \|R\|$. It is easy to see that the matrix A is negative semidefinite and therefore $-\Delta t A$ is positive semidefinite. Using the fact that I and $-\Delta t A$ are both symmetric and positive semidefinite, it can then be shown that $\|(I - \Delta t A)^{-1}\| \leq 1$ for any $\Delta t > 0$. Therefore a sufficient condition for stability in the l^∞-norm is

$$\|I + \Delta t J\|_\infty \leq 1. \tag{3.2.16}$$

Because J is a block diagonal matrix, (3.2.16) is equivalent to

$$\|I + \Delta t J_i\|_\infty \leq 1, \quad \forall i = 1, \dots N. \tag{3.2.17}$$

Dropping the i for simplicity, we then need, for each row j of our matrix:

$$|1 + \Delta t J_{j,j}| + \Delta t \sum_{k=1, k\neq j}^{p+1} |J_{j,k}| \leq 1. \tag{3.2.18}$$

If $J_{j,j} > 0$, then any $\Delta t > 0$ is not a solution. Therefore, we need our model to have $J_{j,j} < 0$. If $\Delta t < -1/J_{j,j}$, then (3.2.18) becomes

$$1 + \Delta t J_{j,j} + \Delta t \sum_{k=1, k\neq j}^{p+1} |J_{j,k}| \leq 1. \tag{3.2.19}$$

$\Rightarrow$

$$J_{j,j} \leq - \sum_{k=1, k\neq j}^{p+1} |J_{j,k}|. \tag{3.2.20}$$

For $\Delta t > -1/J_{j,j}$ we have

$$-1 - \Delta t J_{j,j} + \Delta t \sum_{k=1, k\neq j}^{p+1} |J_{j,k}| \leq 1, \tag{3.2.21}$$

$\Rightarrow$

$$\Delta t \leq \frac{2}{\sum_{k=1}^{p+1} |J_{j,k}|}, \tag{3.2.22}$$

which is equivalent to

$$\Delta t \leq 2/\|J_i\|_\infty. \tag{3.2.23}$$

Hence a sufficient condition for stability for the Forward-Backward Euler scheme is that $\Delta t \leq \dfrac{2}{\max_{i=1,\dots,N} \|J_i\|_\infty}$. We also need our model to satisfy $J_{j,j} < 0$ and (3.2.20) for all $j = 1, \dots, p+1$.

The critical time-step given by this analysis ensures stability in the l^∞-norm as opposed to the L^2 stability obtained above with the Fourier transform, but it is usually too restrictive. In the case of the Beeler-Reuter model, this critical time-step is about 5000 times smaller than its actual numerical value. Therefore, we will only consider the analysis using the Fourier transform and eigenvalues.

3.2.3 Strang splitting

We use the technique presented above to look at the stability of the Strang splitting scheme. Let us start with the CN-RK4 method.

Step 1:

$$\left(\begin{bmatrix} U_\omega^* \\ \mathbf{V}_\omega^* \end{bmatrix} - \begin{bmatrix} U_\omega^n \\ \mathbf{V}_\omega^n \end{bmatrix} \right) \frac{1}{\Delta t/2} = \frac{1}{6}(K_1 + 2K_2 + 2K_3 + K_4), \tag{3.2.24}$$

where

$$K_1 = J \begin{bmatrix} U_\omega^n \\ \mathbf{V}_\omega^n \end{bmatrix}$$

$$K_2 = J \left(\begin{bmatrix} U_\omega^n \\ \mathbf{V}_\omega^n \end{bmatrix} + \frac{\Delta t}{4} K_1 \right)$$

$$K_3 = J \left(\begin{bmatrix} U_\omega^n \\ \mathbf{V}_\omega^n \end{bmatrix} + \frac{\Delta t}{4} K_2 \right)$$

$$K_4 = J \left(\begin{bmatrix} U_\omega^n \\ \mathbf{V}_\omega^n \end{bmatrix} + \frac{\Delta t}{2} K_3 \right)$$

Replacing the K_i into (3.2.25), we obtain

$$\begin{bmatrix} U_\omega^* \\ \mathbf{V}_\omega^* \end{bmatrix} = \left(I + \frac{\Delta t}{2} J + \frac{(\Delta t/2)^2}{2} J^2 + \frac{(\Delta t/2)^3}{6} J^3 + \frac{(\Delta t/2)^4}{24} J^4 \right) \begin{bmatrix} U_\omega^n \\ \mathbf{V}_\omega^n \end{bmatrix}, \quad (3.2.25)$$

Step 2:

$$\left(\begin{bmatrix} U_\omega^{**} \\ \mathbf{V}_\omega^{**} \end{bmatrix} - \begin{bmatrix} U_\omega^* \\ \mathbf{V}_\omega^* \end{bmatrix} \right) \frac{1}{\Delta t} = \frac{1}{2} A_\omega \begin{bmatrix} U_\omega^* \\ \mathbf{V}_\omega^* \end{bmatrix} + \frac{1}{2} A_\omega \begin{bmatrix} U_\omega^{**} \\ \mathbf{V}_\omega^{**} \end{bmatrix}, \quad (3.2.26)$$

$\Longleftrightarrow$

$$\begin{bmatrix} U_\omega^{**} \\ \mathbf{V}_\omega^{**} \end{bmatrix} = \left(I - \frac{1}{2} \Delta t A_\omega \right)^{-1} \left(\frac{1}{2} \Delta t A_\omega + I \right) \begin{bmatrix} U_\omega^* \\ \mathbf{V}_\omega^* \end{bmatrix}, \quad (3.2.27)$$

Step 3:

$$\begin{bmatrix} U_\omega^{n+1} \\ \mathbf{V}_\omega^{n+1} \end{bmatrix} = \left(I + \frac{\Delta t}{2} J + \frac{(\Delta t/2)^2}{2} J^2 + \frac{(\Delta t/2)^3}{6} J^3 + \frac{(\Delta t/2)^4}{24} J^4 \right) \begin{bmatrix} U_\omega^{**} \\ \mathbf{V}_\omega^{**} \end{bmatrix}. \quad (3.2.28)$$

Combining (3.2.25), (3.2.27) and (3.2.28) we obtain

$$\begin{aligned} \begin{bmatrix} U_\omega^{n+1} \\ \mathbf{V}_\omega^{n+1} \end{bmatrix} = & \left(I + \frac{\Delta t}{2} J + \frac{(\Delta t/2)^2}{2} J^2 + \frac{(\Delta t/2)^3}{6} J^3 + \frac{(\Delta t/2)^4}{24} J^4 \right) \left(I - \frac{1}{2} \Delta t A_\omega \right)^{-1} \\ & \left(\frac{1}{2} \Delta t A_\omega + I \right) \left(I + \frac{\Delta t}{2} J + \frac{(\Delta t/2)^2}{2} J^2 + \frac{(\Delta t/2)^3}{6} J^3 + \frac{(\Delta t/2)^4}{24} J^4 \right) \begin{bmatrix} U_\omega^n \\ \mathbf{V}_\omega^n \end{bmatrix}. \end{aligned} \quad (3.2.29)$$

We have the stability function

$$\begin{aligned} R(\Delta t, h, \omega) = & \left(I + \frac{\Delta t}{2} J + \frac{(\Delta t/2)^2}{2} J^2 + \frac{(\Delta t/2)^3}{6} J^3 + \frac{(\Delta t/2)^4}{24} J^4 \right) \left(I - \frac{1}{2} \Delta t A_\omega \right)^{-1} \\ & \left(\frac{1}{2} \Delta t A_\omega + I \right) \left(I + \frac{\Delta t}{2} J + \frac{(\Delta t/2)^2}{2} J^2 + \frac{(\Delta t/2)^3}{6} J^3 + \frac{(\Delta t/2)^4}{24} J^4 \right). \end{aligned}$$

For stability we need $\rho(R(\Delta t, h, \omega)) \leq 1$. This condition depends on $\omega \in [0, \frac{2\pi}{h}]$ and therefore we take the value of ω which is the most restrictive for stability. As for the

Forward-Backward Euler scheme, we take $\omega = 0$. A necessary condition for absolute stability is then given by

$$\rho\left(\left(I + \frac{\Delta t}{2}J + \frac{(\Delta t/2)^2}{2}J^2 + \frac{(\Delta t/2)^3}{6}J^3 + \frac{(\Delta t/2)^4}{24}J^4\right)^2\right) \leq 1. \tag{3.2.30}$$

If this matrix is diagonalizable, the eigenvalues of the polynomial of a matrix are equal to the values of the polynomial evaluated at the eigenvalues of the orignal matrix, (3.2.30) becomes

$$\left|\left(1 + \Delta t\lambda_i/2 + \frac{(\Delta t\lambda_i/2)^2}{2} + \frac{(\Delta t\lambda_i/2)^3}{6} + \frac{(\Delta t\lambda_i/2)^4}{24}\right)^2\right| \leq 1, \tag{3.2.31}$$

for all eigenvalue λ_i of J. As seen with the previous methods, and with the contour of the stability region shown in Figure 3.1, the most restrictive eigenvalue will be the most negative. Using this most negative eigenvalue and solving using Newton's method we obtain the critical time-step shown in Tables 3.2, 3.5 and 3.7. As we did for FBE, we can verify numerically that our stability condition $\rho(R(\Delta t^*_{theo}, h, \omega)) \leq 1$ is satisfied for all other choices of $\omega \in [0, \frac{2\pi}{h}]$.

Using the same technique, we look at the CN-RK2 method.

Step 1:

$$\begin{bmatrix} U^*_\omega \\ \mathbf{V}^*_\omega \end{bmatrix} = \left(I + \frac{\Delta t}{2}J + \frac{(\Delta t/2)^2}{2}J^2\right)\begin{bmatrix} U^n_\omega \\ \mathbf{V}^n_\omega \end{bmatrix}. \tag{3.2.32}$$

Step 2:

$$\begin{bmatrix} U^{**}_\omega \\ \mathbf{V}^{**}_\omega \end{bmatrix} = \left(I - \frac{1}{2}\Delta t A_\omega\right)^{-1}\left(\frac{1}{2}\Delta t A_\omega + I\right)\begin{bmatrix} U^*_\omega \\ \mathbf{V}^*_\omega \end{bmatrix}. \tag{3.2.33}$$

Step 3:

$$\begin{bmatrix} U^{n+1}_\omega \\ \mathbf{V}^{n+1}_\omega \end{bmatrix} = \left(I + \frac{\Delta t}{2}J + \frac{(\Delta t/2)^2}{2}J^2\right)\begin{bmatrix} U^{**}_\omega \\ \mathbf{V}^{**}_\omega \end{bmatrix}. \tag{3.2.34}$$

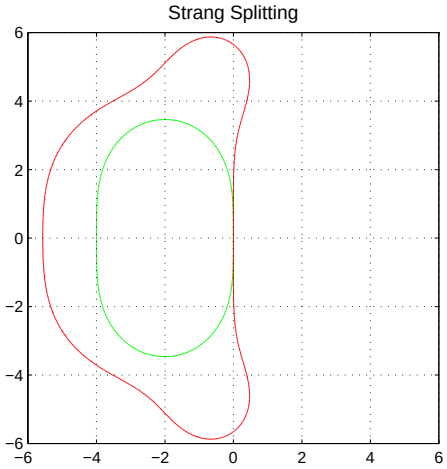

Figure 3.1: Stability regions of CN-RK4 (red) and CN-RK2 (green)

Combining (3.2.32), (3.2.33) and (3.2.34) we obtain

$$\begin{bmatrix} U_\omega^{n+1} \\ \mathbf{V}_\omega^{n+1} \end{bmatrix} = \left(I + \frac{\Delta t}{2} J + \frac{(\Delta t/2)^2}{2} J^2 \right) \left(I - \frac{1}{2} \Delta t A_\omega \right)^{-1} \left(\frac{1}{2} \Delta t A_\omega + I \right) \left(I + \frac{\Delta t}{2} J + \frac{(\Delta t/2)^2}{2} J^2 \right) \begin{bmatrix} U_\omega^{n} \\ \mathbf{V}_\omega^{n} \end{bmatrix}. \quad (3.2.35)$$

For stability we need $\rho(R(\Delta t, h, \omega)) \leq 1$. Again taking $\omega = 0$ because it is the most restrictive wave number for stability, and using the same argument, the stability condition becomes

$$\left| \left(1 + \Delta t \lambda_i / 2 + \frac{(\Delta t \lambda_i / 2)^2}{2} \right)^2 \right| \leq 1, \quad (3.2.36)$$

for all eigenvalue λ_i of J. Similarly, we solve for the contour and get the critical time-step Δt^*_{theo} in Table 3.2 for the BR model, in Table 3.5 for the MS model and in Table 3.7 for the TNNP model. The stability region is the interior of the contour shown in Figure 3.1. This region in contained in the stability region for the CN-RK4 method.

3.2.4 Second order semi-implicit backward differentiation

Following the approach used previously, we write the SBDF2 scheme as:

$$\left(\frac{3}{2}\begin{bmatrix}U_\omega^{n+1}\\V_\omega^{n+1}\end{bmatrix}-2\begin{bmatrix}U_\omega^{n}\\V_\omega^{n}\end{bmatrix}+\frac{1}{2}\begin{bmatrix}U_\omega^{n-1}\\V_\omega^{n-1}\end{bmatrix}\right)\frac{1}{\Delta t}=J\left(2\begin{bmatrix}U_\omega^{n}\\V_\omega^{n}\end{bmatrix}-\begin{bmatrix}U_\omega^{n-1}\\V_\omega^{n-1}\end{bmatrix}\right)+A_\omega\begin{bmatrix}U_\omega^{n+1}\\V_\omega^{n+1}\end{bmatrix}. \tag{3.2.37}$$

For simplicity, we denote $Y^j=\begin{bmatrix}U_\omega^{j}\\V_\omega^{j}\end{bmatrix}$ for all $j\in\mathbb{Z}$. Equation (3.2.37) becomes

$$(\frac{3}{2}I-A_\omega)Y^{n+1}-(2I+2\Delta tJ)Y^n+(\frac{1}{2}I+\Delta tJ)Y^{n-1}=0. \tag{3.2.38}$$

As done for the previous semi-implicit methods, we take $\omega=0$. We will later justify numerically that this value is the most restrictive for the stability of this method. Equation (3.2.38) becomes

$$\frac{3}{2}Y^{n+1}-(2I+2\Delta tJ)Y^n+(\frac{1}{2}I+\Delta tJ)Y^{n-1}=0. \tag{3.2.39}$$

Equation (3.2.39) is solved similarly to the Lagrange's method found in [11]. We set $Y^j=\zeta^jW$, where W is an eigenvector of J with eigenvalue λ. We divide by ζ^{n-1} and obtain the following equation

$$\frac{3}{2}\zeta^2W-(2I+2\Delta tJ)\zeta W+(\frac{1}{2}I+\Delta tJ)W=0. \tag{3.2.40}$$

$\Rightarrow$

$$\frac{3}{2}\zeta^2-(2+2\Delta t\lambda)\zeta+(\frac{1}{2}+\Delta t\lambda)=0. \tag{3.2.41}$$

Equation (3.2.39) has stable solutions iff for any eigenvalue λ, all roots of (3.2.41)satisfy $|\zeta_j(\lambda\Delta t)|\leq 1$, and additional multiple roots must satisfy $|\zeta_j(\lambda\Delta t)|<1$ [12]. The stability region of the method is defined as

$$S=\left\{\mu\in\mathbb{C}\,;\ \begin{array}{l}\text{all roots } \zeta_j(\mu) \text{ of (3.2.41) satisfy } |\zeta_j(\mu)|\leq 1,\\ \text{multiple roots satisfy } |\zeta_j(\mu)|<1\end{array}\right\} \tag{3.2.42}$$

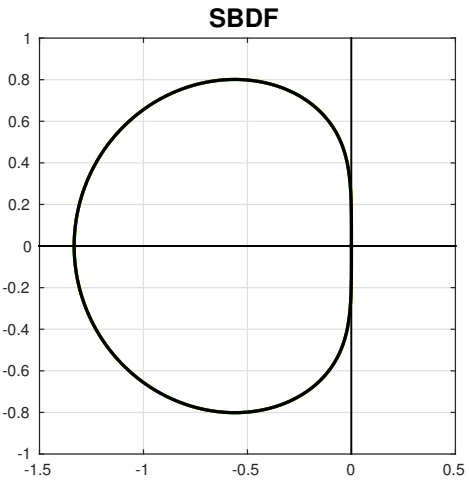

Figure 3.2: Stability Region of the SBDF2 method

For stability of our method we need $\mu = \Delta t\lambda \in S$ for all eigenvalues λ of J. Solving for (3.2.41), we get that our method is stable if

$$\mu = \frac{\frac{-3}{2}\zeta^2 + 2\zeta - \frac{1}{2}}{1 - 2\zeta}$$

for $|\zeta| \leq 1$. We can take $\zeta = e^{i\theta}$ for $0 \leq \theta \leq 2\pi$ to find the contour of the stability region. Hence, the contour of the stability region is

$$\mu = \frac{\frac{-3}{2}e^{2i\theta} + 2e^{i\theta} - \frac{1}{2}}{1 - 2e^{i\theta}}.$$

The contour of the stability region is shown in Figure 3.2. As seen with the previous methods, the most restrictive eigenvalue will be the most negative. Considering only real eigenvalues, we then need $-4/3 \leq \lambda\Delta t \leq 0$ for stability. The critical time-step is then $\Delta t^*_{theo} = \dfrac{-4}{3\lambda_{min}}$ and its value is shown in Table 3.2 for the BR model, in Table 3.7 for the MS model and in Table 3.7 for the TNNP model.

Justification for the choice of ω

We want to verify numerically that for different choices of ω we still have a stable solution when using Δt^*_{theo}. We rewrite equation (3.2.38) such that we have a one-step

method

$$\begin{bmatrix} Y^{n+1} \\ Y^n \end{bmatrix} = \begin{bmatrix} (\frac{3}{2}I - A_\omega)^{-1}(2I + 2\Delta t J) & (\frac{3}{2}I - A_\omega)^{-1}(-\frac{1}{2}I - \Delta t J) \\ I & 0 \end{bmatrix} \begin{bmatrix} Y^n \\ Y^{n-1} \end{bmatrix}. \tag{3.2.43}$$

Let

$$R(\Delta t, h, \omega) = \begin{bmatrix} (\frac{3}{2}I - A_\omega)^{-1}(2I + 2\Delta t J) & (\frac{3}{2}I - A_\omega)^{-1}(-\frac{1}{2}I - \Delta t J) \\ I & 0 \end{bmatrix}$$

be the stability function for the SBDF2 method. The stability condition is $\rho(R(\Delta t, h, \omega)) \leq 1$. We can numerically verify that for different choices of $\omega \in [0, \frac{2\pi}{h}]$, $\rho(R(\Delta t^*_{theo}, h, \omega)) \leq 1$ for J evaluated at any node of the discretized domain. Therefore the stability condition $\Delta t \leq -4/3\lambda_{min}$ is sufficient, at least from numerical verification.

3.2.5 Rush-Larsen

We now look at the stability of the RL-FBE and the RL-CNAB schemes. These differ from the methods previously studied in that the differential equations for the gating variables are solved using a different method than for the membrane potential and the concentrations. The major advantage of using Rush-Larsen methods is to be able to use large time-steps compared to more classical methods. In fact, it can be shown that in the case of most problems in electrophysiology, the region of stability of RL methods of the form (2.3.18) covers the entire negative half plane [24], i.e. A-stability. The stability of the scheme then depends on the methods used to solve the differential equations for the membrane potential and the concentrations. The following discussion on the stability of the Rush-Larsen methods uses heuristic arguments that provide critical time-steps relatively close to those in numerical tests, but this derivation cannot yet be formalized.

Let us first study the unconditional stability of the Rush-Larsen (RL) method using

a scalar problem. The RL method is used for solving problems of the following form

$$\frac{dy}{dt} = a(y)y + b(y), \tag{3.2.44}$$

which gives

$$y^{n+1} = e^{a^n h}\left(y^n + \frac{b^n}{a^n}\right) - \frac{b^n}{a^n}, \tag{3.2.45}$$

where $a^n = a(y^n)$ and $b^n = b(y^n)$.

To study the absolute-stability of this scheme we consider the following problem:

$$\frac{dy}{dt} = \lambda y, \tag{3.2.46}$$

where $\lambda = \lambda_a + \lambda_b < 0$. Here, λ_a represents the $a(y)$ in equation (3.2.44), and $\lambda_b y$ is a linearization of $b(y)$, i.e. $b^n \approx \lambda_b y^n$. Equation (3.2.45) becomes:

$$y^{n+1} = e^{\lambda_a \Delta t}\left(1 + \frac{\lambda_b}{\lambda_a}\right) y^n - \frac{\lambda_b}{\lambda_a} y^n. \tag{3.2.47}$$

The stability condition for this scheme is:

$$-1 \leq e^{\lambda_a \Delta t}\left(1 + \frac{\lambda_b}{\lambda_a}\right) - \frac{\lambda_b}{\lambda_a} \leq 1. \tag{3.2.48}$$

Knowing that $\lambda_b = \lambda - \lambda_a$, we get

$$\frac{\lambda}{\lambda_a} - 2 \leq e^{\lambda_a \Delta t}\frac{\lambda}{\lambda_a} \leq \frac{\lambda}{\lambda_a}. \tag{3.2.49}$$

Assuming $\lambda_a < 0$, we get $\dfrac{\lambda}{\lambda_a} > 0$, which gives us

$$1 - 2\frac{\lambda_a}{\lambda} \leq e^{\lambda_a \Delta t} \leq 1. \tag{3.2.50}$$

Because $\lambda_a < 0$, the right inequality is always satisfied. If $\lambda_a \leq \lambda/2$ then the left inequality is also always satisfied. Therefore, assuming $\lambda_a \leq \lambda/2$, the method is unconditionally stable. Otherwise, if we have $\lambda_a > \lambda/2$, we apply the ln function to (3.2.50) and from the left inequality we obtain the following stability condition for Δt

$$\Delta t \leq \frac{1}{\lambda_a} \ln\left(1 - 2\frac{\lambda_a}{\lambda}\right). \tag{3.2.51}$$

In the case of problems in cardiac electrophysiology, we have $\lambda < 0$, $\lambda_a < 0$ and $\lambda_b > 0$ [24].

Rush-Larsen with Forward-Backward Euler

We now look at the stability of the system. In this case, we use a different scheme for the differential equations of the gating variables and the concentrations. Therefore, we look at the problem in the form (1.2.6)-(1.2.8). As done previously, we linearize around some constant state $(\tilde{u}, \tilde{\mathbf{v}}, \tilde{\mathbf{X}}) = \tilde{Y}$ and take the Fourier transform. We apply the scheme (2.3.3) to the linearized problem. The scheme is identical to Forward-Backward Euler for the membrane potential and concentrations.

$$\left(1 - \sigma\frac{2\cos\omega h - 2}{h^2}\right) U_\omega^{n+1} = U_\omega^n + \Delta t \begin{bmatrix} -\frac{\partial I_{ion}}{\partial u}(\tilde{Y}) & -\frac{\partial I_{ion}}{\partial \mathbf{v}}(\tilde{Y}) & -\frac{\partial I_{ion}}{\partial \mathbf{X}}(\tilde{Y}) \end{bmatrix} \begin{bmatrix} U_\omega^n \\ V_\omega^n \\ X_\omega^n \end{bmatrix}, \tag{3.2.52}$$

$$X_\omega^{n+1} = X_\omega^n + \Delta t \begin{bmatrix} \frac{\partial \mathbf{g}}{\partial u}(\tilde{Y}) & \frac{\partial \mathbf{g}}{\partial \mathbf{v}}(\tilde{Y}) & \frac{\partial \mathbf{g}}{\partial \mathbf{X}}(\tilde{Y}) \end{bmatrix} \begin{bmatrix} U_\omega^n \\ V_\omega^n \\ X_\omega^n \end{bmatrix}. \tag{3.2.53}$$

The ODEs for the gating variables are solved with the first order Rush-Larsen method. As opposed to previously studied methods, it is not a linear multistep method. To facilitate the absolute stability analysis and because the Rush-Larsen method is A-stable, we will consider an A-stable one-step linear method for V, which can be written as

$$V_\omega^{n+1} = R(\Delta t, \omega) V_\omega^n, \tag{3.2.54}$$

where $\rho(R(\Delta t, \omega)) \leq 1$, for any $\Delta t > 0$. We then consider the most restrictive case of $\rho(R(\Delta t, \omega)) = 1$. This is the case when $R(\Delta t, \omega)) = I$, which is equivalent to setting

$$V_\omega^{n+1} = V_\omega^n. \tag{3.2.55}$$

As for the previous semi-implicit methods, we take $\omega = 0$ because it is the choice that is most restrictive for stability. Combining (3.2.52)-(3.2.55), we have

$$Y^{n+1} = (I + \Delta t J_{RL})Y^n, \tag{3.2.56}$$

where

$$J_{RL} = \begin{bmatrix} -\frac{\partial I_{ion}}{\partial u}(\tilde{Y}) & -\frac{\partial I_{ion}}{\partial \mathbf{v}}(\tilde{Y}) & -\frac{\partial I_{ion}}{\partial \mathbf{X}}(\tilde{Y}) \\ 0 & 0 & 0 \\ \frac{\partial \mathbf{g}}{\partial u}(\tilde{Y}) & \frac{\partial \mathbf{g}}{\partial \mathbf{v}}(\tilde{Y}) & \frac{\partial \mathbf{g}}{\partial \mathbf{X}}(\tilde{Y}) \end{bmatrix}. \tag{3.2.57}$$

As we did before for the Euler methods, the stability condition for this scheme is $\Delta t \leq -2/\lambda_{min}$, where λ_{min} is the most negative eigenvalue of J_{RL}. The critical time-step, $\Delta t^*_{theo} = -2/\lambda_{min}$ is shown in Table 3.2 for the BR model and in Table 3.7 for the TNNP model.

Similarly, for the RL-CNAB method, we get from the stability analysis of the Adams Bashforth method, $\Delta t^*_{theo} = -1/\lambda_{min}$.

3.2.6 Deferred Correction

Due to the more complex nature of the third order deferred correction scheme, we cannot easily find a stability condition using absolute stability analysis. However, the numerically observed critical time-step is very close to the one from the Forward-Backward Euler method.

3.3 Numerical results

In this section, we compare the critical time-steps obtained through our absolute stability analysis of from the previous section to those observed numerically in the case of the BR, MS and TNNP models.

The numerical results in the 1D case were obtained with the 1D problem (1.2.2)-(1.2.4). We use a spatial domain of length 100 cm, discretized by equally spaced

nodes $x_i = ih$, $h = 1/N$ or its equivalent for the nondimensionalized MS model. We used a final time T of 400 ms for the BR model, 350 ms for the MS model and 300 ms for the TNNP model. The values of the parameters for the monodomain model are shown in Tables B.1, B.2 and B.3 in Appendix B. The initial values for the different variables at time $t = 0$ are the resting values shown in Tables B.4 and B.5. The applied current stimulation is the C^∞ function given by

$$I_{app}(\mathbf{x}, t) = \begin{cases} 50 \exp\left(1 - \dfrac{1}{1-(t-1.5)^2}\right) \exp\left(1 - \dfrac{1}{1-(2(x-0.5))^2}\right) & \text{if } 0 < x < 1, \\ & 0.5 < t < 2.5, \\ 0 & \text{otherwise.} \end{cases} \tag{3.3.1}$$

All simulations for the 1D case were made with MATLAB.

We also tested the stability of the BR model in the 2D case using an implementation of the methods written in Fortran90 called SIMCA. [1] For the 2D simulations, we used a 1cm $\times$ 1cm square domain discretized with 3432 grid points and a final time T of 16 ms. The applied current stimulation is given in SIMCA and is a C^1 function of x and t consisting of a modified cosine on a chosen subset of grid points and zero elsewhere. It is given by

$$I_{app}(\mathbf{x}, t) = \begin{cases} 50 \dfrac{1+\cos(\pi r)}{2} \dfrac{1+\cos(\pi \tau)}{2} & \text{if } 0 \le r \le 1,\ 0 \le \tau \le 1, \\ 0 & \text{otherwise,} \end{cases} \tag{3.3.2}$$

where $r = |\mathbf{x} - \mathbf{x_0}|/r_0$ with $\mathbf{x_0}$ the point at the center of a circular simulation zone and r_0, the radius of this zone. Similarly, $\tau = (t - t_0)/\tau_0$ sets a stimulation interval in time starting at $t = t_0$ with duration τ_0.

For each method, the theoretical critical time-step Δt^*_{theo} is determined by the stability conditions from the previous section and the most negative eigenvalue of the Jacobian of the ionic model used, λ_{min}, which is calculated on the domain as explained

[1] Charles Pierre, Université de Pau et des Pays de l'Adour, Laboratoires de Mathématiques et de leurs Applications

in the previous section. The numerically observed critical time-step Δt^* is the largest possible time-step for which the numerical solution remains bounded. When using a relatively large time-step with the more stable Rush-Larsen methods, the solution remains bounded but its shape degenerates and does not represent the real shape of the wave. Therefore, for the RL methods the value of Δt^* is taken as the largest possible time-step for which the potential wave does not degenerate.

Remark: Computing the Jacobian for discontinuous models.
The functions I, F and G of the BR model are all continuous and differentiable with respect to each variable. We thus found the Jacobian analytically and evaluated it numerically. Other ionic models have discontinuities which prevent us from defining a Jacobian for those points. To avoid deriving expressions on both sides of the discontinuities, we decided to approximate the Jacobian numerically, for e.g. by using MATLAB's `numjac` function. For example, several functions in the ODEs for the gating variables in the TNNP model have discontinuities at $u = -40$ and the function in the ODE of the gating variable of the MS model has a discontinuity at $u = u_{gate}$. These are null sets of the phase space and in general we will not observe these discontinuities when approximating the Jacobian or solving the differential equations numerically. However, if we approximate the Jacobian at the discontinuity point or close enough to be below the tolerance for `numjac`, we obtain extremely large eigenvalues. These discontinuities might become a problem for extremely small time-steps because it becomes more likely to have values of u very close to the singularities. This has not been a problem for our simulations.

3.3.1 Beeler-Reuter

The most negative eigenvalue of the Jacobian for the Beeler-Reuter model is $\lambda_{min} = -81.782$. For this model, this value can be obtained by evaluating the Jacobian at

the resting values found in Table B.4. Note that one can find the most negative eigenvalues of the Jacobian of 37 different ionic models in [30] and [22]. The authors obtained a value of -82.0 for the BR model. The theoretical critical time-steps for methods studied above are shown in Table 3.2 and the numerically observed critical time-steps in Table 3.1 for the 1D case and in Table 3.3 for the 2D case. We can see that for all methods except the RL methods, the critical time-steps obtained numerically are very close to those obtained through absolute stability analysis. In the case of the RL methods, the critical time-steps are similar for RL-FBE, but Δt^* is approximately half of Δt^*_{theo} for RL-CNAB. This is likely a consequence of our stability analysis using heuristic arguments.

For the Strang splitting methods, we observe smaller Δt^* in 2D compared to their 1D equivalent. Otherwise, the analysis done in the last section for the 1D monodomain model seems to apply for the semi-implicit methods in the 2D case. Due to the more complex nature of the Finite Element method for higher dimensions, the analysis of explicit methods such as FE will depend on the meshing technique used.

We observe that the critical time-steps of the FE and FBE methods are initially the same for $h = 0.0625$, but as h gets smaller FE becomes less stable with a critical time-step of order h^2. This indicates that the use of explicit methods to solve the BR model is only justifiable for very coarse meshes.

The RL methods are the most stable of all the methods studied. The RL-CNAB has a Δt^* more than three times larger than the one of the next most stable method, CN-RK4. This method has a Δt^* slightly larger than the one for CN-RK2, which reflects the fact that the stability region of CN-RK2 is included in the stability region of CN-RK4. The Δt^* for the SBDF2 method is three to four times smaller than the ones for the Strang splitting methods and 50% smaller than the one for FBE. These relations between the Δt^* of the linear multistep methods, i.e. excluding the RL methods, are the same for all the ionic models used.

As mentioned in the last section, the critical time-step for the DC3 method is very

close to the value for FBE: it is slightly smaller.

Table 3.1: Size of Δt^* for the numerical methods used with BR model

Methods	h=0.062 500	h=0.031 250	h=0.015 625
2nd Order SBDF	0.016 848	0.016 848	0.016 848
Strang Splitting (CN-RK4)	0.071 480	0.071 480	0.071 480
Strang Splitting (CN-RK2)	0.050 289	0.050 289	0.050 289
Forward Euler	0.025 373	0.020 094	0.005 063 9
Forward-Backward Euler	0.025 373	0.025 373	0.025 373
RL-CNAB	0.235 29	0.235 29	0.235 29
RL-FBE	>0.800 00	>0.800 00	>0.800 00
DC3	0.024 465	0.024 465	0.024 465

Table 3.2: Size of Δt^*_{theo} for the numerical methods used with BR model

Methods	h=0.062 500	h=0.031 250	h=0.015 625
2nd Order SBDF	0.016 304	0.016 304	0.016 304
Strang Splitting (CN-RK4)	0.068 115	0.068 115	0.068 115
Strang Splitting (CN-RK2)	0.048 911	0.048 911	0.048 911
Forward Euler	0.024 455	0.020 238	0.005 066 2
Forward-Backward Euler	0.024 455	0.024 455	0.024 455
RL-CNAB	0.423 42	0.423 42	0.423 42
RL-FBE	0.846 83	0.846 83	0.846 83

3.3.2 Mitchell-Schaeffer

For the MS model, the most negative eigenvalue obtained is $\lambda_{min} = -2.6651$. The theoretical critical time-steps for the methods studied are shown in Table 3.5 and the

Table 3.3: Size of Δt^* for the numerical methods used with BR model in 2D

Methods	Δt^*
2^{nd} Order SBDF	0.016 131
Strang Splitting (CN-RK4)	0.059 566
Strang Splitting (CN-RK2)	0.044 933
Forward-Backward Euler	0.024 242
RL-CNAB	0.200 00

numerically observed critical time-steps in Table 3.4. We can see that for all methods Δt^* is very close to Δt^*_{theo}.

Because the MS model is not very stiff, we see the stability of the FE method depends on the size of h even for relatively large values. This indicates the necessity for taking the diffusion implicitly when solving less stiff models such as the MS model.

We also observe that the CN-RK4 method requires a slightly smaller time-step than Δt^*_{theo}. This is most likely resulting from the oscillations caused by the use of the Crank-Nicolson method [36]. These oscillations can be observed for large time-steps for both Strang splitting methods.

For the MS model, the Strang splitting methods are the most stable of all the methods studied. They have a Δt^* three to four times larger than for the SBDF2 method.

The DC3 and FBE methods have the same numerically observed critical time-step.

3.3.3 ten Tuscher-Noble-Noble-Panfilov

For the TNNP model, the most negative eigenvalue obtained is $\lambda_{min} = -1191.7$. The value given in [30] and [22] is -1170. The theoretical critical time-steps for the methods studied are shown in Table 3.7 and the numerically observed critical time-steps in Table 3.7. As for the BR model, we can see that for all methods except the RL methods, the critical time-steps obtained numerically are very close to those

Table 3.4: Size of Δt^* for the numerical methods used with MS model

Methods	h=1	h=0.500 00	h=0.250 00
2nd Order SBDF	0.526 32	0.526 32	0.526 32
Strang Splitting (CN-RK4)	1.8519	1.8519	1.8519
Strang Splitting (CN-RK2)	1.5217	1.5217	1.5217
Forward Euler	0.122 70	0.035 874	0.008 940 5
Forward-Backward Euler	0.769 23	0.769 23	0.769 23

Table 3.5: Size of Δt^*_{theo} for the numerical methods used with MS model

Methods	h=1	h=0.500 00	h=0.250 00
2nd Order SBDF	0.500 29	0.500 29	0.500 29
Strang Splitting (CN-RK4)	2.0902	2.0902	2.0902
Strang Splitting (CN-RK2)	1.5009	1.5009	1.5009
Forward Euler	0.111 79	0.033 672	0.008 872 8
Forward-Backward Euler	0.750 43	0.750 43	0.750 43
DC3	0.750 43	0.750 43	0.750 43

obtained through absolute stability analysis. In the case of the RL methods, Δt^* is approximately half of Δt^*_{theo} for RL-CNAB and two thirds for RL-FBE.

For the Forward Euler scheme, Δt^* is the same for $h = 0.0625, 0.03125, 0.015625$. For smaller values of h, we begin to see a dependence of Δt^* on h^2, as shown in Table 3.8. These results indicate that for very stiff models such as the TNNP model, the use of fully explicit methods could still be acceptable, except on finer spatial meshes. The RL methods are the most stable of all the methods studied. The RL-CNAB has a Δt^* almost twenty times larger than the next most stable method, CN-RK4.

As for the BR model, the DC3 method has a Δt^* slightly smaller than the one for FBE.

Table 3.6: Size of Δt^* for the numerical methods used with TNNP model

Methods	h=0.062 500	h=0.031 250	h=0.015 625
2nd Order SBDF	0.001 134 8	0.001 134 8	0.001 134 8
Strang Splitting (CN-RK4)	0.004 836 2	0.004 836 2	0.004 836 2
Strang Splitting (CN-RK2)	0.003 456 2	0.003 456 2	0.003 456 2
Forward Euler	0.001 704 4	0.001 704 4	0.001 704 4
Forward-Backward Euler	0.001 704 4	0.001 704 4	0.001 704 4
RL-CNAB	0.091 380	0.091 380	0.091 380
RL-FBE	>0.600 00	>0.600 00	>0.600 00
DC3	0.001 686 9	0.001 686 9	0.001 686 9

Table 3.7: Size of Δt^*_{theo} for the numerical methods used with TNNP model

Methods	h=0.062 500	h=0.031 250	h=0.015 625
2nd Order SBDF	0.001 118 9	0.001 118 9	0.001 118 9
Strang Splitting (CN-RK4)	0.004 674 6	0.004 674 6	0.004 674 6
Strang Splitting (CN-RK2)	0.003 356 6	0.003 356 6	0.003 356 6
Forward Euler	0.001 678 3	0.001 678 3	0.001 678 3
Forward-Backward Euler	0.001 678 3	0.001 678 3	0.001 678 3
RL-CNAB	0.206 12	0.206 12	0.206 12
RL-FBE	0.412 23	0.412 23	0.412 23

As expected from the analysis of the last sections, for all ionic models, only the fully explicit Forward Euler's method has a critical time-step depending on the size of h. Indeed, as h gets smaller we see that Δt^* is eventually proportional to h^2. Looking at the Jacobian for large h, we see that the minimum eigenvalue comes from the gating variables. For the FE method, it then requires small values of h for the eigenvalue coming from the second derivative term in the PDE for the transmembrane

Table 3.8: Size of Δt^* for the Forward Euler's method with TNNP model

h	Δt^*	Δt^*_{theo}
0.015625	0.001 704 4	0.001 678 3
0.0078125	0.001 267 7	0.001 266 9
0.00390625	0.000 316 82	0.000 316 76

potential to become the most negative. This leads to the conclusion that for very stiff models such as the TNNP model, taking the diffusion implicitly is only necessary for very fine meshes.

For all models, the semi-implicit method which requires the smallest time-step is the SBDF2 method.

3.3.4 Energy method

We will now find the critical time-steps from the stability conditions obtained for the FE and SBDF2 methods in Section 3.1. As we have mentionned, the stability conditions from that section are too restrictive. We will show this for the MS model. The nondimensionalization parameter of the MS model is $\sigma = 3.4585$. We take the $\max(L_I, L_f)$ as the spectral radius of the Jacobian. For FE, the first condition gives a critical time-step $\Delta t^*_{theo'} = 1/C = -1/\lambda_{min} = 0.37522$. This condition is not dependent on h and the obtained critical time-step is half of the value of Δt^* for the semi-implicit FBE.

The second condition for FE gives us a critical time-step $\Delta t^*_{theo''} = h^2/6\sigma$. These values, compared to the numerically observed values, are shown in Table 4.9. $\Delta t^*_{theo''}$ is approximately a third of the values observed numerically.

For the SBDF2 method, we obtain $\Delta t^*_{theo'} = 1/C = -1/12\lambda_{min} = 0.031268$. It is 16 times smaller than the value observed numerically.

Table 3.9: Size of Δt^* for the FE method with MS model

h	Δt^*	$\Delta t^*_{theo''}$
1	0.122 70	0.048 191
0.5	0.035 874	0.012 048
0.25	0.008 940 5	0.003 011 9

Chapter 4

Accuracy of the numerical methods

In this chapter, we investigate the accuracy of the different time-stepping methods when solving the monodomain model coupled with the three ionic models studied. We start by conducting a convergence test for each method to verify if they have the correct rate of convergence. We will also compare the accuracy of the methods relative to the size of the time-step used. Afterwards, we will compare the accuracy of the methods relative to the computational time needed to run the simulations.

4.1 Convergence Tests

In Chapter 2, we defined the order of a numerical method using the error between the exact and numerical solutions. Let y and $y_{\Delta t}$ be, respectively, the exact and numerical solutions obtained using the time-step Δt of the problem (2.1.1). Then if $\max_{0\leq n\leq N_t} \|y(t_n) - y_{\Delta t}(t_n)\| = \mathcal{O}(\Delta t^p)$, the method is said of order p. In our case, we will look at the order of the error at the final time T, $\|y(T) - y_{\Delta t}(T)\|$, which bounds below the numerical error over time steps. As is the case for the problem studied, the analytical solution y is not always available. A good substitute for the convergence test is to calculate a solution using a very small time-step, which we call a reference

solution.

The convergence of the different methods will now be investigated. We intend to study the convergence of time-stepping methods on a given spatial mesh. We split the error in space and time given that their order may not be the same and study the error in time only. We have

$$\|u - u_{h,\Delta t}\| \leq \|u - u_h\| + \|u_h - u_{h,\Delta t}\| = \mathcal{O}(h^p + \Delta t^q), \tag{4.1.1}$$

where u is the exact solution for the transmembrane potential, u_h is the solution of the problem semi-discretized in space and $u_{h,\Delta t}$ is the solution of the fully discretized problem. Because we want to study the convergence of the time-stepping methods, we will only consider the error between the solutions of the semi-discretized and fully discretized problems. We test convergence with respect to the following errors:

$$e_{L^2} = \|u_{h,\Delta t}(T) - u_h(T)\|_{L^2}, \tag{4.1.2}$$

$$e_{H^1} = |u_{h,\Delta t}(T) - u_h(T)|_{H^1}, \tag{4.1.3}$$

$$e_c = |c_{h,\Delta t} - c_h|, \tag{4.1.4}$$

$$e_{\hat{T}} = |\hat{T}_{h,\Delta t} - \hat{T}_h|, \tag{4.1.5}$$

where u_h is a semi-discretized reference solution for the transmembrane potential, which is calculated using a very small time-step. The norms used are discrete approximations of the $L^2(\Omega)$ norm and $H^1(\Omega)$ seminorm using Simpson's rule. The solutions $u_{h,\Delta t}$ are calculated with the same spatial mesh and at the same final time T as the reference solution, but using larger values for Δt. We denote by $c_{h,\Delta t}$ the wave velocity and $\hat{T}_{h,\Delta t}$ the depolarization time, i.e. the time at which a given point of the domain reaches a super-threshold value of the transmembrane potential. c_h and $\hat{T}_h$ are the wave velocity and depolarization time of the reference solution u_h. The wave velocity is defined by $c = (x_2 - x_1)/(T_2 - \hat{T})$ where $\hat{T}$ is the time when

the depolarization front of the potential wave first passes through a chosen node x_1 and T_2 when it passes through a second chosen node x_2. The wave front passes through a point x_i during the time-step from t_n to $t_{n+1} = t_n + \Delta t$ if $u(x_i, t_n) < \hat{u}$, but $u(x_i, t_{n+1}) \geq \hat{u}$ for some chosen value $\hat{u}$ on the wave front. For a better approximation of $\hat{T}$ and T_2, we use linear interpolation and define

$$T_i = t_n + \Delta t \frac{\hat{u} - u(x_i, t_n)}{u(x_i, t_{n+1}) - u(x_i, t_n)}.$$

Assuming that an error is proportional to Δt^α, the estimated convergence rate is calculated with

$$\alpha = \frac{\log(|e_1/e_2|)}{\log(\Delta t_1/\Delta t_2)}, \tag{4.1.6}$$

where Δt_1 and Δt_2 are consecutive time-steps in a sequence of decreasing time-steps, and e_1 and e_2 are the corresponding errors. All solutions are calculated using the same stimulation current and constants from Section 3.3. The largest Δt used for each method is taken close to the critical time-step found in the previous chapter, except for the RL methods where we use a starting time-step similar to the other methods, that is when the RL methods start to show their asymptotic behaviour.
The order of convergence in the L^2 norm and H^1 seminorm are expected to be the same because the numerical solutions $u_{h,\Delta t}$ are in the same finite-dimensional space and thus the equivalence of norms applies.

4.1.1 Beeler-Reuter with 1D monodomain

The reference solution u_h for the BR model is computed using SBDF2 on a domain of length 100 *cm* discretized in space with 1600 nodes and computed at time $T = 400\,\text{ms}$ with $\Delta t = 1/125\,000\,\text{ms}$. The plot of the reference solution is shown in Figure 4.1. To determine c and $\hat{T}$, we take $\hat{u} = -30\,\text{mV}$, $x_1 = 20\,\text{cm}$ and $x_2 = 50\,\text{cm}$. When testing for the DC3 method, SBDF2 did not give a reference solution accurate enough for

assessing the third order of converence therefore we used DC3 instead of SBDF2 to calculate a reference solution with $\Delta t = 1/75\,000$ ms. The results of the convergence tests for the 1D BR model are shown in Tables 4.1 to 4.8. For each table, the first column is a sequence of time-steps in ms of decreasing size. The next columns are the errors as defined in (4.1.2)-(4.1.5), followed by their respective observed rate of convergence as defined in (4.1.6). In each Table, we indicate in red the asymptotic zone of the approximate rates of convergence, i.e. when the rate of convergence is very close to the order of convergence of the method. We observe that $\hat{T}$ and c have larger asymptotic zones than the errors in L^2 norm and H^1 seminorm and that e_{L^2} has a larger asymptotic zone than e_{H^1}.

For the 1D BR model, all of the methods studied showed their expected asymptotic order of convergence. For the first order methods, we can observe that for the same Δt, FE is almost twice as accurate as its semi-implicit version, FBE, and both methods are significantly more accurate than RL-FBE. It takes very small values of Δt for RL-FBE to reach order of convergence one.

For the second-order methods, we can observe that for the same Δt, CN-RK4 is about ten times more accurate than CN-RK2, which in turn is about two times more accurate than SBDF2. This last method is two times more accurate than RL-CNAB. For the DC3 method, it takes very small values of Δt before it exhibits its correct order of convergence. Even for $\Delta t = 1/5120$, the error is still larger than what we would obtain by using a second-order method.

In the case of the BR model, all methods studied converge as expected.

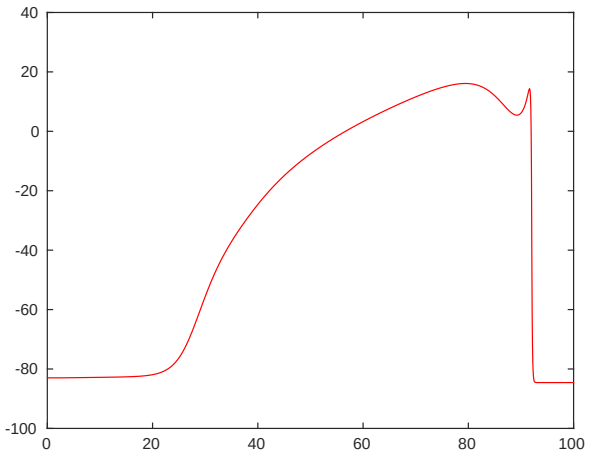

Figure 4.1: Reference solution for the membrane potential at time $T = 400\,\text{ms}$ for the BR model in 1D plotted as a function of $x \in [0, 100]$.

Table 4.1: Errors for FE with BR model at fixed $h = 1/16$

Δt	e_{L^2}	α_{L^2}	e_{H^1}	α_{H^1}	e_c	α_c	$e_{\hat{T}}$	$\alpha_{\hat{T}}$
1/40	57.7	-	221	-	1.17e−3	-	4.83e−1	-
1/80	50.0	0.207	215	0.0417	9.69e−4	0.271	3.72e−1	0.379
1/160	28.8	0.796	170	0.336	4.89e−4	0.987	1.87e−1	0.991
1/320	15.3	0.916	109	0.643	2.46e−4	0.994	9.38e−2	0.995
1/640	7.79	0.970	58.3	0.900	1.23e−4	0.997	4.70e−2	0.998
1/1280	3.85	1.02	29.4	0.989	6.16e−5	0.998	2.35e−2	0.999
1/2560	1.94	0.993	15.6	0.918	3.08e−5	0.999	1.18e−2	0.999

Table 4.2: Errors for FBE with BR model at fixed $h = 1/16$

Δt	e_{L^2}	α_{L^2}	e_{H^1}	α_{H^1}	e_c	α_c	$e_{\hat{T}}$	$\alpha_{\hat{T}}$
1/40	103	-	225	-	−3.15e−3	-	1.22	-
1/80	80.3	0.360	225	−0.001 52	−2.01e−3	0.652	7.56e−1	0.693
1/160	51.6	0.637	215	0.0672	−1.02e−3	0.976	3.83e−1	0.982
1/320	29.9	0.788	173	0.310	−5.15e−4	0.988	1.93e−1	0.991
1/640	15.9	0.908	112	0.627	−2.58e−4	0.994	9.66e−2	0.995
1/1280	8.15	0.967	61.2	0.875	−1.29e−4	0.997	4.84e−2	0.998
1/2560	4.03	1.02	30.7	0.998	−6.48e−5	0.998	2.42e−2	0.999

Table 4.3: Errors for RL-FBE with BR model at fixed $h = 1/16$

Δt	e_{L^2}	α_{L^2}	e_{H^1}	α_{H^1}	e_c	α_c	$e_{\hat{T}}$	$\alpha_{\hat{T}}$
1/30	231	-	220	-	−1.54e−2	-	6.10	-
1/60	164	0.499	221	−0.008 51	−7.91e−3	0.961	3.03	1.01
1/120	117	0.486	223	−0.0139	−4.01e−3	0.980	1.51	1.00
1/240	80.5	0.536	225	−0.009 80	−2.02e−3	0.990	7.53e−1	1.00
1/480	51.3	0.650	215	0.0675	−1.01e−3	0.995	3.76e−1	1.00
1/960	29.5	0.797	172	0.319	−5.08e−4	0.997	1.88e−1	1.00
1/1920	15.7	0.915	111	0.636	−2.54e−4	0.999	9.39e−2	1.00

Table 4.4: Errors for SBDF2 with BR model at fixed $h = 1/16$

Δt	e_{L^2}	α_{L^2}	e_{H^1}	α_{H^1}	e_c	α_c	$e_{\hat{T}}$	$\alpha_{\hat{T}}$
1/60	34.9	-	187	-	6.27e−4	-	−2.15e−1	-
1/120	1.83	4.25	14.8	3.66	−2.93e−5	4.42	1.10e−2	4.29
1/240	0.458	2.00	3.84	1.95	−7.24e−6	2.02	2.71e−3	2.02
1/480	0.114	2.00	0.966	1.99	−1.80e−6	2.01	6.72e−4	2.01
1/960	0.0285	2.00	0.241	2.00	−4.48e−7	2.00	1.68e−4	2.00
1/1920	0.007 11	2.00	0.0603	2.00	−1.12e−7	2.00	4.18e−5	2.00
1/3840	0.001 77	2.00	0.0150	2.00	−2.79e−8	2.00	1.04e−5	2.00

Table 4.5: Errors for RL-CNAB with BR model at fixed $h = 1/16$

Δt	e_{L^2}	α_{L^2}	e_{H^1}	α_{H^1}	e_c	α_c	$e_{\hat{T}}$	$\alpha_{\hat{T}}$
1/30	70.8	-	225	-	−1.63e−3	-	6.08e−1	-
1/60	24.7	1.52	154	0.545	−4.15e−4	1.98	1.54e−1	1.98
1/120	6.52	1.92	48.5	1.67	−1.04e−4	1.99	3.86e−2	2.00
1/240	1.63	2.00	13.2	1.88	−2.61e−5	2.00	9.65e−3	2.00
1/480	0.412	1.99	3.45	1.94	−6.52e−6	2.00	2.41e−3	2.00
1/960	0.103	2.00	0.872	1.98	−1.63e−6	2.00	6.03e−4	2.00
1/1920	0.0258	2.00	0.219	2.00	−4.08e−7	2.00	1.51e−4	2.00

Table 4.6: Errors for CN-RK4 with BR model at fixed $h = 1/16$

Δt	e_{L^2}	α_{L^2}	e_{H^1}	α_{H^1}	e_c	α_c	$e_{\hat{T}}$	$\alpha_{\hat{T}}$
1/15	4.70	-	35.0	-	−7.59e−5	-	2.59e−2	-
1/30	1.50	1.65	12.1	1.54	−2.41e−5	1.66	8.30e−3	1.64
1/60	0.392	1.94	3.24	1.90	−6.24e−6	1.95	2.14e−3	1.96
1/120	0.0989	1.99	0.823	1.98	−1.57e−6	1.99	5.39e−4	1.99
1/240	0.0248	2.00	0.207	1.99	−3.93e−7	2.00	1.35e−4	2.00
1/480	0.006 20	2.00	0.0517	2.00	−9.81e−8	2.00	3.38e−5	2.00
1/960	0.001 55	2.00	0.0129	2.00	−2.45e−8	2.00	8.44e−6	2.00

Table 4.7: Errors for CN-RK2 with BR model at fixed $h = 1/16$

Δt	e_{L^2}	α_{L^2}	e_{H^1}	α_{H^1}	e_c	α_c	$e_{\hat{T}}$	$\alpha_{\hat{T}}$
1/20	30.2	-	173	-	5.51e−4	-	−1.54e−1	-
1/40	6.84	2.14	50.9	1.77	−1.10e−4	2.33	3.96e−2	1.96
1/80	1.59	2.10	12.9	1.98	−2.55e−5	2.10	9.23e−3	2.10
1/160	0.388	2.03	3.25	1.99	−6.17e−6	2.05	2.23e−3	2.05
1/320	0.0958	2.02	0.807	2.01	−1.52e−6	2.02	5.49e−4	2.02
1/640	0.0238	2.01	0.201	2.01	−3.76e−7	2.01	1.36e−4	2.01
1/1280	0.005 92	2.01	0.0500	2.01	−9.37e−8	2.01	3.39e−5	2.01

Table 4.8: Errors for DC3 with BR model at fixed $h = 1/16$

Δt	e_{L^2}	α_{L^2}	e_{H^1}	α_{H^1}	e_c	α_c	$e_{\hat{T}}$	$\alpha_{\hat{T}}$
1/80	68.3	-	219	-	−1.62e−3	-	2.43e−1	-
1/160	34.3	0.992	180	0.285	−5.81e−4	1.48	5.69e−2	2.10
1/320	12.0	1.52	83.6	1.11	−1.54e−4	1.91	1.01e−2	2.49
1/640	2.89	2.05	20.7	2.01	−2.99e−5	2.37	1.50e−3	2.75
1/1280	0.535	2.43	4.02	2.37	−4.66e−6	2.68	2.03e−4	2.88
1/2560	0.0811	2.72	0.615	2.71	−6.47e−7	2.85	2.64e−5	2.94
1/5120	0.0111	2.87	0.0843	2.87	−8.49e−8	2.93	3.35e−6	2.98

4.1.2 Mitchell-Schaeffer with 1D monodomain

The reference solution for the MS model is computed using SBDF2 on a domain of length 800 discretized in space with 800 nodes and computed at time 350 with $\Delta t = 7/60000$. The plot of the reference solution is shown in Figure 4.2. To determine c and $\hat{T}$, we take $\hat{u} = 0.5$, $x_1 = 50$ and $x_2 = 80$.For the convergence of the DC3 method, we used the same method instead of SBDF2 to calculate a reference solution with $\Delta t = 7/60000$. The results of the convergence tests for the 1D MS model are shown in Tables 4.9 to 4.14.

For the MS model, all methods studied showed their expected asymptotic order of convergence, but the Strang splitting methods lose their order when Δt gets smaller. This could be caused by the discontinuities in the ionic model. For the first order methods, we can observe that for the same Δt, FBE is slightly more accurate than its fully explicit version, FE.

For second-order methods, we can observe that for the same Δt, CN-RK4 is almost ten times more accurate than CN-RK2, which in turn is about twenty times more accurate than SBDF2. For the DC3 method, it takes very small values of Δt before it exhibits its correct order of convergence. Even for $\Delta t = 7/6144$, its error is still larger than what we would obtain by using a good second-order method.

In the case of the MS model, the rate of convergence of the Strang-splitting deteriorates for smaller values of Δt. The other methods exhibit the correct rate of convergence.

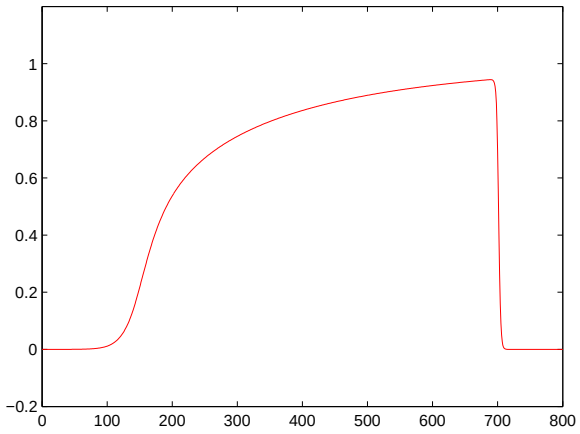

Figure 4.2: Reference solution for the membrane potential at time $T = 350\,\text{ms}$ for the MS model in 1D plotted as a function of $x \in [0, 800]$.

Table 4.9: Errors for FE with MS model at fixed $h = 1$

Δt	e_{L^2}	α_{L^2}	e_{H^1}	α_{H^1}	e_c	α_c	$e_{\hat{T}}$	$\alpha_{\hat{T}}$
7/96	3.19	-	0.439	-	−4.13e−2	-	6.70	-
7/192	1.97	0.692	0.403	0.122	−2.09e−2	0.980	3.36	0.995
7/384	1.09	0.851	0.281	0.519	−1.05e−2	0.990	1.68	0.997
7/768	0.567	0.950	0.158	0.836	−5.29e−3	0.995	8.42e−1	0.999
7/1536	0.286	0.985	0.0813	0.954	−2.65e−3	0.997	4.21e−1	0.999
7/3072	0.144	0.996	0.0410	0.988	−1.33e−3	0.999	2.11e−1	1.000
7/6144	0.0719	0.998	0.0205	0.997	−6.63e−4	0.999	1.05e−1	0.999

Table 4.10: Errors for FBE with MS model at fixed $h = 1$

Δt	e_{L^2}	α_{L^2}	e_{H^1}	α_{H^1}	e_c	α_c	$e_{\hat{T}}$	$\alpha_{\hat{T}}$
7/24	5.81	-	0.436	-	−1.16e−1	-	1.98e1	-
7/48	4.10	0.502	0.437	−0.002 01	−6.24e−2	0.895	1.03e1	0.937
7/96	2.72	0.591	0.432	0.0162	−3.24e−2	0.943	5.29	0.966
7/192	1.64	0.735	0.368	0.230	−1.66e−2	0.970	2.68	0.981
7/384	0.886	0.884	0.236	0.639	−8.38e−3	0.984	1.35	0.990
7/768	0.455	0.961	0.128	0.883	−4.21e−3	0.993	6.76e−1	0.995
7/1536	0.230	0.987	0.0656	0.966	−2.11e−3	0.994	3.39e−1	0.997

Table 4.11: Errors for SBDF2 with MS model at fixed $h = 1$

Δt	e_{L^2}	α_{L^2}	e_{H^1}	α_{H^1}	e_c	α_c	$e_{\hat{T}}$	$\alpha_{\hat{T}}$
7/24	3.42	-	0.441	-	−4.61e−2	-	7.52	-
7/48	1.26	1.44	0.314	0.489	−1.23e−2	1.90	1.97	1.93
7/96	0.335	1.91	0.0950	1.72	−3.09e−3	1.99	4.93e−1	2.00
7/192	0.0837	2.00	0.0239	1.99	−7.77e−4	1.99	1.23e−1	2.01
7/384	0.0206	2.02	0.005 91	2.02	−1.92e−4	2.02	3.02e−2	2.02
7/768	0.005 11	2.02	0.001 46	2.01	−4.87e−5	1.98	7.49e−3	2.01
7/1536	0.001 34	1.94	0.000 382	1.94	−1.21e−5	2.01	1.96e−3	1.94

Table 4.12: Errors for CN-RK4 with MS model at fixed $h = 1$

Δt	e_{L^2}	α_{L^2}	e_{H^1}	α_{H^1}	e_c	α_c	$e_{\hat{T}}$	$\alpha_{\hat{T}}$
7/12	1.30e−1	-	3.46e−2	-	−1.27e−3	-	1.77e−1	-
7/24	4.11e−2	1.66	1.11e−2	1.64	−4.01e−4	1.67	5.76e−2	1.62
7/48	1.04e−2	1.98	2.82e−3	1.98	−1.05e−4	1.93	1.48e−2	1.96
7/96	2.55e−3	2.03	7.00e−4	2.01	−2.32e−5	2.18	3.63e−3	2.03
7/192	7.80e−4	1.71	2.14e−4	1.71	−6.50e−6	1.83	1.07e−3	1.75
7/384	2.56e−4	1.61	7.01e−5	1.61	−1.53e−6	2.09	3.54e−4	1.60
7/768	1.20e−4	1.10	3.19e−5	1.14	3.04e−7	2.33	1.62e−4	1.13

Table 4.13: Errors for CN-RK2 with MS model at fixed $h = 1$

Δt	e_{L^2}	α_{L^2}	e_{H^1}	α_{H^1}	e_c	α_c	$e_{\hat{T}}$	$\alpha_{\hat{T}}$
7/12	6.38e−1	-	1.73e−1	-	−6.01e−3	-	9.38e−1	-
7/24	1.78e−1	1.84	5.00e−2	1.79	−1.64e−3	1.87	2.59e−1	1.86
7/48	4.59e−2	1.96	1.29e−2	1.95	−4.22e−4	1.96	6.68e−2	1.96
7/96	1.16e−2	1.98	3.26e−3	1.98	−1.05e−4	2.01	1.70e−2	1.97
7/192	3.08e−3	1.92	8.69e−4	1.91	−2.83e−5	1.89	4.39e−3	1.96
7/384	8.29e−4	1.89	2.33e−4	1.90	−5.58e−6	2.34	1.23e−3	1.83
7/768	2.78e−4	1.58	7.75e−5	1.58	−1.04e−6	2.42	4.25e−4	1.54

Table 4.14: Errors for DC3 with MS model at fixed $h = 1$

Δt	e_{L^2}	α_{L^2}	e_{H^1}	α_{H^1}	e_c	α_c	$e_{\hat{T}}$	$\alpha_{\hat{T}}$
7/24	5.41	-	4.27e−1	-	−1.16e−1	-	1.72e1	-
7/48	3.57	0.599	4.32e−1	−0.0155	−6.08e−2	0.928	8.01	1.10
7/96	1.97	0.860	3.98e−1	0.117	−2.84e−2	1.10	3.24	1.30
7/192	7.57e−1	1.38	2.05e−1	0.958	−1.07e−2	1.41	1.06	1.61
7/384	1.98e−1	1.93	5.65e−2	1.86	−2.93e−3	1.87	2.60e−1	2.03
7/768	3.81e−2	2.38	1.09e−2	2.37	−5.79e−4	2.34	4.83e−2	2.43
7/1536	6.00e−3	2.67	1.72e−3	2.67	−9.18e−5	2.66	7.46e−3	2.69
7/3072	8.34e−4	2.85	2.39e−4	2.85	−1.26e−5	2.86	1.03e−3	2.86
7/6144	1.13e−4	2.88	3.25e−5	2.88	−1.69e−6	2.90	1.39e−4	2.89

4.1.3 ten Tuscher-Noble-Noble-Panfilov with 1D monodomain

Due to the complexity and stability requirements of the TNNP model, we study the convergence of the methods on a smaller spatial domain. The domain will be too small for the whole wave to develop, but still includes the depolarization wavefront. The reference solution is computed using SBDF2 on a domain of length 5 cm discretized in space with 160 nodes and computed at time 12 ms with $\Delta t = 6e-7$. The reference solution is shown in Figure 4.3. To determine c and $\hat{T}$, we take $\hat{u} = -30\,\text{mV}$, $x_1 = 1\,\text{cm}$ and $x_2 = 2.5\,\text{cm}$. For the convergence of the DC3 method, we used the same method instead of SBDF2 to calculate a reference solution with $\Delta t = 7.5e-7$. The results of the convergence tests for the 1D TNNP model are shown in Tables 4.15 to 4.22.

For the TNNP model, all methods studied showed their expected asymptotic order of convergence, but the convergence of the RL-CNAB and DC3 methods is erratic when Δt gets very small. The solution obtained with the RL-CNAB method first converges to the reference solution obtained with SBDF2 then the convergence stagnates when Δt gets very small. The L^2 error between the solutions seems to stabilize around $e_{L^2} = 0.00115$, which is less than 0.0012% of the reference solution's L^2 norm. The DC3 method shows similar trends with third order convergence that deteriorates when Δt gets very small. This is likely due to the difficulty of computing a reference solution at such small level of errors. Another reason for these strange behaviors with very small time-steps might be caused by the discontinuities in the right-hand-side of the ODEs for the TNNP model. For a small time-step, it is more likely that the numerical solution falls close to a discontinuity at some of the grid points. Before its order of convergence deteriorates, the DC3 method is actually more accurate than the second-order methods.

It is important to note that the TNNP model is very stiff and that is why we use

very small time-steps for the convergence tests. In practice, it is not relevant to have errors as small as those obtained for the smallest time-steps used for the modelling error is then larger then the numerical error.

For the first order methods, we can observe that for the same Δt, FE is slightly more accurate than its semi-implicit version, FBE, and both methods are more than twice as accurate as RL-FBE.

For second-order methods, the Δt used are not the same, but one can easily see that for the same Δt, CN-RK4 is almost two times more accurate than CN-RK2, which in turn is about two times more accurate than SBDF2. This last method is about two times more accurate than RL-CNAB.

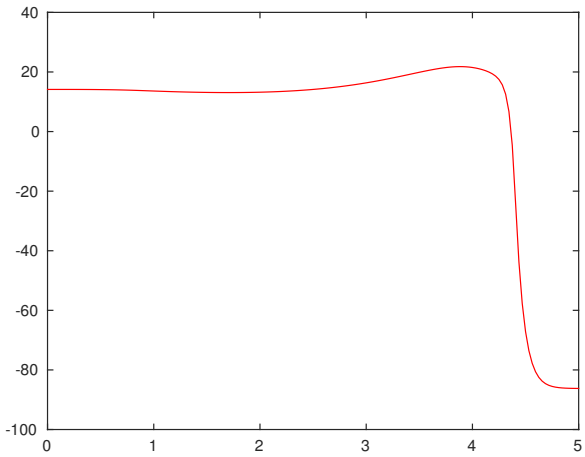

Figure 4.3: Reference solution for the membrane potential at time $T = 12\,\text{ms}$ for the TNNP model in 1D plotted as a function of $x \in [0, 5]$.

Table 4.15: Errors for FE with TNNP model at fixed $h = 1/32$

Δt	e_{L^2}	α_{L^2}	e_{H^1}	α_{H^1}	e_c	α_c	$e_{\hat{T}}$	$\alpha_{\hat{T}}$
1/600	1.18	-	14.0	-	−4.89e−4	-	4.21e−3	-
1/1200	0.592	0.997	7.02	1.00	−2.45e−4	0.997	2.11e−3	0.999
1/2400	0.296	0.999	3.51	1.00	−1.23e−4	0.998	1.05e−3	1.000
1/4800	0.148	1.000	1.75	1.00	−6.13e−5	0.999	5.27e−4	1.000
1/9600	0.0741	1.000	0.875	1.00	−3.07e−5	1.000	2.63e−4	1.000
1/19200	0.0370	1.000	0.437	1.00	−1.53e−5	1.000	1.32e−4	1.000
1/38400	0.0185	1.00	0.219	1.00	−7.67e−6	1.00	6.58e−5	1.00

Table 4.16: Errors for FBE with TNNP model at fixed $h = 1/32$

Δt	e_{L^2}	α_{L^2}	e_{H^1}	α_{H^1}	e_c	α_c	$e_{\hat{T}}$	$\alpha_{\hat{T}}$
1/600	1.23	-	15.4	-	−5.34e−4	-	4.16e−3	-
1/1200	0.620	0.992	7.75	0.993	−2.68e−4	0.993	2.08e−3	0.997
1/2400	0.311	0.997	3.88	0.999	−1.34e−4	0.996	1.04e−3	0.999
1/4800	0.156	0.998	1.94	1.00	−6.73e−5	0.998	5.22e−4	0.999
1/9600	0.0778	0.999	0.969	1.00	−3.37e−5	0.999	2.61e−4	1.000
1/19200	0.0389	1.000	0.484	1.00	−1.68e−5	0.999	1.31e−4	1.000
1/38400	0.0195	1.00	0.242	1.00	−8.42e−6	1.000	6.53e−5	1.000

Table 4.17: Errors for RL-FBE with TNNP model at fixed $h = 1/32$

Δt	e_{L^2}	α_{L^2}	e_{H^41}	α_{H^1}	e_c	α_c	$e_{\hat{T}}$	$\alpha_{\hat{T}}$
1/300	5.42	-	62.8	-	−2.48e−3	-	1.41e−2	-
1/600	2.75	0.978	33.0	0.929	−1.25e−3	0.987	7.09e−3	0.996
1/1200	1.39	0.986	16.8	0.969	−6.29e−4	0.994	3.55e−3	0.998
1/2400	0.697	0.995	8.46	0.994	−3.15e−4	0.997	1.78e−3	0.999
1/4800	0.349	0.999	4.23	1.00	−1.58e−4	0.999	8.88e−4	1.00
1/9600	0.174	1.00	2.11	1.00	−7.87e−5	1.00	4.44e−4	1.00
1/19200	0.0871	1.00	1.05	1.00	−3.93e−5	1.00	2.22e−4	1.00

Table 4.18: Errors for SBDF2 with TNNP model at fixed $h = 1/32$

Δt	e_{L^2}	α_{L^2}	e_{H^1}	α_{H^1}	e_c	α_c	$e_{\hat{T}}$	$\alpha_{\hat{T}}$
1/900	4.73e−3	-	5.31e−2	-	−1.92e−6	-	1.99e−5	-
1/1800	1.17e−3	2.02	1.31e−2	2.02	−4.74e−7	2.02	4.92e−6	2.01
1/3600	2.94e−4	1.99	3.30e−3	1.98	−1.23e−7	1.94	1.18e−6	2.06
1/7200	7.66e−5	1.94	8.56e−4	1.95	−3.36e−8	1.87	2.95e−7	2.00
1/14400	1.97e−5	1.96	2.22e−4	1.95	−9e−9	1.90	7.16e−8	2.04
1/28800	3.68e−6	2.42	4.19e−5	2.40	−8.60e−10	3.39	2.07e−8	1.79
1/57600	9.03e−7	2.02	9.41e−6	2.15	−1.94e−11	5.47	4e−9	2.36

Table 4.19: Errors for RL-CNAB with TNNP model at fixed $h = 1/32$

Δt	e_{L^2}	α_{L^2}	e_{H^1}	α_{H^1}	e_c	α_c	$e_{\hat{T}}$	$\alpha_{\hat{T}}$
3/640	0.152	-	1.78	-	−6.68e−5	-	4.44e−4	-
3/1280	0.0381	1.99	0.447	2.00	−1.67e−5	2.00	1.10e−4	2.01
3/2560	0.009 46	2.01	0.110	2.02	−4.07e−6	2.03	2.73e−5	2.02
3/5120	0.002 49	1.92	0.0260	2.08	−9.13e−7	2.16	6.45e−6	2.08
3/10240	0.001 21	1.04	0.004 79	2.44	−1.12e−7	3.03	1.12e−6	2.53
3/20480	0.001 14	0.0826	0.001 16	2.04	9.41e−8	0.246	−2.42e−7	2.21
3/40960	0.001 15	−0.0147	0.002 24	−0.941	1.42e−7	−0.593	−5.98e−7	−1.31

Table 4.20: Errors for CN-RK4 with TNNP model at fixed $h = 1/32$

Δt	e_{L^2}	α_{L^2}	e_{H^1}	α_{H^1}	e_c	α_c	$e_{\hat{T}}$	$\alpha_{\hat{T}}$
3/640	1.71e−2	-	1.77e−1	-	−8.47e−6	-	2.92e−6	-
3/1280	4.27e−3	2.00	4.43e−2	2.00	−2.09e−6	2.02	5.90e−8	5.63
3/2560	1.07e−3	2.00	1.11e−2	2.00	−5.40e−7	1.95	2.43e−7	−2.04
3/5120	2.75e−4	1.96	2.88e−3	1.94	−1.44e−7	1.91	1.38e−7	0.821
3/10240	7.60e−5	1.86	7.91e−4	1.86	−4.05e−8	1.83	3.37e−8	2.03
3/20480	1.81e−5	2.07	1.82e−4	2.12	−8.28e−9	2.29	8e−9	2.08
3/40960	4.53e−6	2.00	5.07e−5	1.84	−2.84e−9	1.55	−1.75e−9	2.20

Table 4.21: Errors for CN-RK2 with TNNP model at fixed $h = 1/32$

Δt	e_{L^2}	α_{L^2}	e_{H^1}	α_{H^1}	e_c	α_c	$e_{\hat{T}}$	$\alpha_{\hat{T}}$
1/300	1.23e−2	-	1.31e−1	-	−6.11e−6	-	7.26e−6	-
1/600	3.09e−3	2.00	3.27e−2	2.00	−1.53e−6	2.00	1.77e−6	2.04
1/1200	7.74e−4	2.00	8.24e−3	1.99	−3.76e−7	2.02	5.13e−7	1.79
1/2400	1.96e−4	1.98	2.07e−3	1.99	−9.49e−8	1.99	1.00e−7	2.35
1/4800	4.81e−5	2.02	5.02e−4	2.04	−2.53e−8	1.91	6.82e−9	3.88
1/9600	1.01e−5	2.25	1.08e−4	2.21	−4.70e−9	2.43	4.72e−9	0.530
1/19200	1.64e−6	2.63	1.60e−5	2.76	−1.09e−9	2.10	7.60e−10	2.64

Table 4.22: Errors for DC3 with TNNP model at fixed $h = 1/32$

Δt	e_{L^2}	α_{L^2}	e_{H^1}	α_{H^1}	e_c	α_c	$e_{\hat{T}}$	$\alpha_{\hat{T}}$
1/600	3.96e−3	-	4.51e−2	-	−1.05e−6	-	−2.99e−7	-
1/1200	5.13e−4	2.95	5.90e−3	2.93	−1.45e−7	2.85	−1.74e−7	0.783
1/2400	6.59e−5	2.96	7.53e−4	2.97	−2.41e−8	2.59	−8.69e−8	0.998
1/4800	8.50e−6	2.95	8.94e−5	3.08	−3e−9	3.00	−3.05e−8	1.51
1/9600	3.98e−7	4.42	9.86e−6	3.18	−6.65e−12	8.82	−6.90e−9	2.14
1/19200	5.22e−7	−0.390	7.74e−6	0.349	2.82e−10	−5.41	4.31e−10	4.00
1/38400	4.03e−7	0.373	5.69e−6	0.443	1.94e−10	0.542	4e−9	−3.21

4.1.4 Beeler-Reuter with 2D monodomain

In the 2D case, the reference solution is computed using SBDF2 on a 1cm × 1cm square domain discretized with an unstructured mesh of 3432 points (roughly of size 59×59) and computed at time $T = 16\,\text{ms}$ with $\Delta t = 5.98e-6$. The reference solution is shown in Figure 4.4. The results of the convergence tests for the 2D BR model are shown in Tables 4.23 to 4.27. We present errors and convergence rates in L^2 norm and H^1 seminorm only; wave velocity and depolarization time were not considered. We also only considered a subset of the methods studied in the 1D case.

As for the 1D BR model, all of the methods studied showed their expected asymptotic order of convergence for the 2D case. For the same Δt, CN-RK4 is about five times more accurate than CN-RK2, which in turn is almost four times more accurate than SBDF2. This last method and RL-CNAB have approximately the same accuracy. The first-order FBE method is significantly less accurate than the higher-order methods.

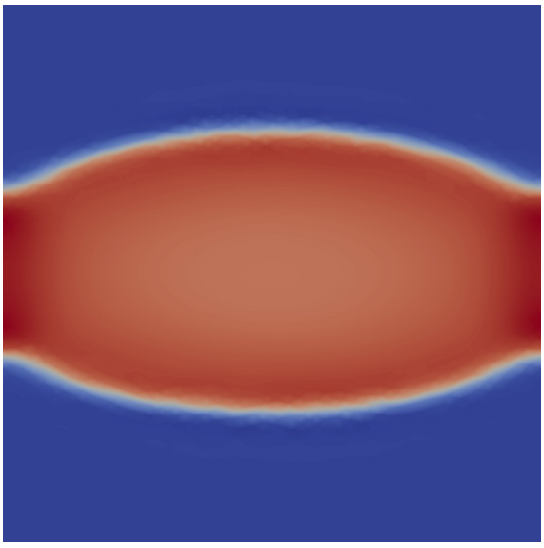

Figure 4.4: Reference solution for the membrane potential at time $T = 16\,\text{ms}$ for the BR model in 2D plotted as a function of $x \in [0,1] \times [0,1]$.

Table 4.23: Errors for FBE with BR model in 2D

Δt	e_{L^2}	α_{L^2}	e_{H^1}	α_{H^1}
1.23×10^{-2}	1.59	-	9.35×10^{1}	-
6.13×10^{-3}	8.04×10^{-1}	0.980	4.75×10^{1}	0.977
3.06×10^{-3}	4.04×10^{-1}	0.991	2.39×10^{1}	0.991
1.53×10^{-3}	2.03×10^{-1}	0.996	1.20×10^{1}	0.996
7.66×10^{-4}	1.02×10^{-1}	0.998	6.00	0.998
3.83×10^{-4}	5.08×10^{-2}	0.999	3.00	0.999
1.91×10^{-4}	2.54×10^{-2}	0.999	1.50	1.000

Table 4.24: Errors for SBDF2 with BR model in 2D

Δt	e_{L^2}	α_{L^2}	e_{H^1}	α_{H^1}
1.23×10^{-2}	5.02×10^{-2}	-	2.78	-
6.13×10^{-3}	1.27×10^{-2}	1.99	6.98×10^{-1}	1.99
3.06×10^{-3}	3.18×10^{-3}	1.99	1.75×10^{-1}	2.00
1.53×10^{-3}	7.95×10^{-4}	2.00	4.38×10^{-2}	2.00
7.66×10^{-4}	1.99×10^{-4}	2.00	1.10×10^{-2}	2.00
3.83×10^{-4}	4.99×10^{-5}	2.00	2.74×10^{-3}	2.00
1.91×10^{-4}	1.22×10^{-5}	2.03	6.78×10^{-4}	2.02

Table 4.25: Errors for RL-CNAB with BR model in 2D

Δt	e_{L^2}	α_{L^2}	e_{H^1}	α_{H^1}
4.89×10^{-2}	7.47×10^{-1}	-	4.15×10^{1}	-
2.45×10^{-2}	1.90×10^{-1}	1.98	1.05×10^{1}	1.98
1.22×10^{-2}	4.77×10^{-2}	1.99	2.64	1.99
6.12×10^{-3}	1.19×10^{-2}	2.00	6.61×10^{-1}	2.00
3.06×10^{-3}	2.99×10^{-3}	2.00	1.65×10^{-1}	2.00
1.53×10^{-3}	7.47×10^{-4}	2.00	4.13×10^{-2}	2.00
7.65×10^{-4}	1.87×10^{-4}	2.00	1.03×10^{-2}	2.00

Table 4.26: Errors for CN-RK2 with BR model in 2D

Δt	e_{L^2}	α_{L^2}	e_{H^1}	α_{H^1}
2.45×10^{-2}	5.81×10^{-2}	-	2.98	-
1.23×10^{-2}	1.32×10^{-2}	2.14	6.73×10^{-1}	2.15
6.13×10^{-3}	3.15×10^{-3}	2.06	1.60×10^{-1}	2.07
3.06×10^{-3}	7.71×10^{-4}	2.03	3.92×10^{-2}	2.03
1.53×10^{-3}	1.91×10^{-4}	2.02	9.68×10^{-3}	2.02
7.66×10^{-4}	4.74×10^{-5}	2.01	2.41×10^{-3}	2.01
3.83×10^{-4}	1.18×10^{-5}	2.00	6.00×10^{-4}	2.00

Table 4.27: Errors for CN-RK4 with BR model in 2D

Δt	e_{L^2}	α_{L^2}	e_{H^1}	α_{H^1}
4.89×10^{-2}	4.98×10^{-2}	-	3.90	-
2.45×10^{-2}	1.20×10^{-2}	2.05	9.56×10^{-1}	2.03
1.22×10^{-2}	2.98×10^{-3}	2.01	2.38×10^{-1}	2.00
6.12×10^{-3}	7.32×10^{-4}	2.00	5.95×10^{-2}	2.00
3.06×10^{-3}	1.80×10^{-4}	2.00	1.49×10^{-2}	2.00
1.53×10^{-3}	5.64×10^{-5}	2.00	3.72×10^{-3}	2.00
7.65×10^{-4}	1.14×10^{-5}	2.00	9.30×10^{-4}	2.00

4.2 CPU performance of the numerical methods

In the last sections, we compared the accuracy of the methods for a fixed number of time-steps. However, this does not take into account the difference of computational cost for an iteration depending on the method used.

We now compare the accuracy of the different numerical methods studied with respect to the computational time of the simulation. We use the same spatial domain and spatial discretization as for the convergence tests. 2D simulations were run on an HP Z420 Worksation with 4 Intel Xeon 2.80 GHz processors and 7.7 GB of RAM and 1D simulations were run on an ALIENWARE X51-R2 with a Intel Core i7-4770 3.40 GHz processor and 7.89 GB of RAM. The L^2 norm and H^1 seminorm for the different reference solutions are, respectively, 519.6 and 155.1 for the 1D BR model, 19.00 and 0.3154 for the 1D MS model, 96.26 and 201.8 for the 1D TNNP model, 61.77 and 655.1 for the 2D BR model. For each model, we choose a target value of the L^2 error on the numerical solution, e_{L^2}, and for each method, we find the time-step needed to achieve this level of accuracy and the computational time of the simulation. Results are shown for the different methods in Tables 4.28 to 4.32. Each table contains the L^2 error, e_{L^2}, close enough to the chosen target error, with its corresponding H^1 error, e_{H^1}, and size of the time-step Δt, the total CPU time of the simulation and the average CPU time per iteration. We indicate in the title of each table the relative size of the error e_{L^2} with respect to the L^2 norm of the reference solution.

For the 1D BR model, we see that the most efficient method is the SBDF2 method. It is almost twice as fast as the next most efficient method, CN-RK4, which is itself faster than the CN-RK2 method. For each time-step of the SBDF2 method, we only have to compute the functions of the ionic model once, as opposed to a time-step for the Strang-splitting methods where we compute them four or eight times for CN-RK2 and CN-RK4, respectively. As for the DC3 method, the functions are computed only

three times per time-step, but it takes small values of Δt for the method to reach a higher rate of convergence. We observe that our third-order method, DC3, is not efficient compared to the second-order methods. Indeed, it takes more than twenty times more CPU time to compute the solution than the SBDF2 method for the chosen error. By choosing very small errors, we expect the DC3 method to eventually surpass the second-order methods. At the chosen error e_{L^2}, the DC3 method has not yet entered its asymptotic zone.

We only tested the computational time of the FBE method at 5% L^2 error because it would take extremely small time-steps to reach the level of error that we chose for the other methods. It is clear that for 0.5% error, the FBE method is not efficient compared to high-order methods.

The computational time per time-step of the SBDF2 method is only slightly higher than for the FBE method and thus there is no advantage in terms of efficiency to using the first-order method. One time-step of CN-RK4 is two times more costly than for CN-RK2, which is almost four times more costly than the SBDF2 method. This is directly related to the number of computations of the ionic functions. One time-step of DC3 is more than three times the cost of an SBDF2 time-step, which also relates to the computation of the ionic functions, but there is additional cost due to the more complex nature of the DC3 method. These relations extend to the different models studied. Even though the RL-CNAB also only has one computation of the ionic functions per time-step, the use of the exponential function makes an iteration for this method slightly more expensive than for the SBDF2 method. This impact is lessened in the case of the TNNP model because the computation of the ionic functions takes up most of the computational cost, due to the large number of variables in the TNNP model.

For the 1D MS model, we choose a slightly smaller relative error because the computational times are smaller and if they are too small than we have issues with their

Table 4.28: CPU time of the numerical methods for the BR model in 1D for 0.5% L^2 error

Method	e_{L^2}	e_{H^1}	Δt	CPU time (s)	CPU/time-step (ms)
SBDF2	2.590	18.67	0.010 00	69.58	1.739
RL-CNAB	2.604	18.74	0.005 333	173.9	2.318
CN-RK4	2.594	18.53	0.045 45	114.3	12.99
CN-RK2	2.591	18.63	0.016 00	159.9	6.396
FBE	25.97	153.0	0.002 694	248.2	1.672
DC3	2.606	18.76	0.001 493	1637	6.106

repeatability. We still choose an error small enough for most methods to be in their asymptotic zones. We see that the most efficient method is the CN-RK4 method, followed closely by the CN-RK2 method. The CN-RK4 method is almost twice as efficient as the SBDF2 method. The RL-CNAB follows slightly after. We observe that our third-order method, DC3, is not efficient compared to the second-order methods. Indeed, it takes almost eight times more CPU time to compute the solution than the SBDF2 method for the chosen error.

We only tested the computational time of the FBE method at 1% L^2 error because it would take extremely small time-steps to reach the error we chose for the other methods. It is easy to figure that for 0.1% error, the FBE method is not efficient compared to high-order methods.

For the 1D TNNP model, most methods require very small time-steps to have stable solutions. Therefore the relative error is very small compared to those of the previous models. We see that the most efficient method is, as for BR, the SBDF2 method. For 0.005% L^2 error, it is nearly two times faster than the next most efficient

Table 4.29: CPU time of the numerical methods for the MS model in 1D for 0.1% L^2 error

Method	e_{L^2}	e_{H^1}	Δt	CPU time (s)	CPU/time-step (ms)
SBDF2	0.018 98	0.005 431	0.017 28	5.772	0.2850
CN-RK4	0.018 84	0.005 091	0.1971	3.276	1.845
CN-RK2	0.018 94	0.005 329	0.092 72	3.572	0.9463
FBE	0.1897	0.054 32	0.003 763	25.37	0.2728
DC3	0.019 03	0.005 461	0.006 972	44.66	0.8897

second-order method, RL-CNAB, more than twice as fast as the CN-RK2 method, and three times faster than the CN-RK4 method. Because the convergence of the RL-CNAB method stagnates (see Table 4.19) and the DC3 method is slow to reach its correct rate of convergence, there is no value of e_{L^2} for which both methods are in their asymptotic zone. This is why we need two tables with different values of L^2 relative error, one to compare with RL-CNAB and one with DC3. We observe that the DC3 method is actually the second most efficient method for the lower 0.001% L^2 error.

We only tested the computational time of the FBE method at 0.05% L^2 error because it would take extremely small time-steps to reach the error we chose for the other methods. Because of the stability restraints of the other methods, only the RL-CNAB can have an error as high as 0.05%. It would be easy to show that when RL-CNAB is in its asymptotic zone, it is more efficient than FBE. It is also easy to figure that for 0.005% error, the FBE method is not efficient compared to high-order methods.

For the 2D case with the BR model, we chose a smaller relative error because the simulations were faster due to the smaller domain and the fact that the computational

Table 4.30: CPU time of the numerical methods for the TNNP model in 1D for 0.005% L^2 error

Method	e_{L^2}	e_{H^1}	Δt	CPU time (s)	CPU/time-step (ms)
SBDF2	0.004 800	0.053 86	0.001 116	6.178	0.5747
RL-CNAB	0.004 802	0.054 67	0.000 834 5	10.78	0.7496
CN-RK4	0.004 785	0.049 52	0.002 487	20.79	4.310
CN-RK2	0.004 797	0.050 84	0.002 078	12.54	2.172
FBE	0.047 96	0.5968	6.417×10^{-5}	106.8	0.5710

Table 4.31: CPU time of the numerical methods for the TNNP model in 1D for 0.001% L^2 error

Method	e_{L^2}	e_{H^1}	Δt	CPU time (s)	CPU/time-step (ms)
SBDF2	0.000 958 7	0.010 75	0.000 502 4	14.40	0.6028
CN-RK4	0.000 958 7	0.009 969	0.001 091	47.58	4.325
CN-RK2	0.000 960 5	0.010 18	0.000 928 1	29.75	2.301
DC3	0.000 961 6	0.010 99	0.001 026	25.51	2.180

code is written in Fortran instead of MATLAB as in 1D. If the CPU times were too small, we had issues with their repeatability. As opposed to 1D case, the most efficient method for the 2D BR model is not the SBDF2 method, but rather the CN-RK4 method, as shown in Table 4.32. It is a bit more efficient than the CN-RK2, which in turn is more efficient than the SBDF2 method. The RL-CNAB is more than four times slower than the CN-RK4 method. As usual, the FBE method is not efficient. The difference in the performance of the methods between the 1D and 2D cases is most likely due to the difference in the programming language for the computational codes. We used MATLAB in the 1D case and because the MATLAB language is a weakly typed interpreted programming language, it tends to work better with shorter, easily

implemented methods, such as SBDF2. On the other hand, we use the strongly typed compiled programming language Fortran90, which does perform well in the case of longer codes. However, the SIMCA Fortran90 implementation used for the 2D simulations is written in a way that impacts the computational cost per iteration because additional information is calculated at each iteration. The SBDF2 and RL-CNAB methods do require more iterations than the Strang splitting methods.

Table 4.32: CPU time of the numerical methods for the BR model in 2D for 0.008% L^2 error

Method	e_{L^2}	e_{H^1}	Δt	CPU time (s)	CPU/time-step (ms)
SBDF2	0.005 012	0.2762	0.003 850	15.86	3.816
RL-CNAB	0.005 001	0.2771	0.001 985	47.79	5.929
CN-RK4	0.005 002	0.3975	0.015 79	10.88	10.74
CN-RK2	0.005 008	0.2550	0.007 678	12.90	6.189
FBE	0.050 30	2.971	0.000 375 0	150.4	3.526

Chapter 5

Conclusion and Future Work

In this thesis, we investigated the impact of ionic model complexity and stiffness on finding an efficient numerical solution through a variety of time-stepping methods. The most simple model that we tested is the MS model. Its stiffness is low and hence there is no need for very stable methods to be used. The most accurate method to solve this problem is Strang splitting using Runge-Kutta methods (RK2 or RK4) to solve the ionic model and the reaction part of the monodomain model, and Crank-Nicholson for the diffusion part of the monodomain model. Due to the nature of operator splitting methods, they become less competitive when solving more complex models. Indeed, for each time-step, we do two RK substeps which for the second order RK method implies computing the ionic currents four times, and for the original fourth order RK, eight times. As the models get more complex and realistic, computing the ionic currents takes up most of the computational cost of the algorithms. Therefore, multi-step methods such as SBDF2 become more efficient with respect to computational time when solving more complex models because the ionic currents are only computed once per time-step. For the Beeler-Reuter model, both the SBDF2 and the Strang splitting methods are the most efficient. The easier implementation of the SBDF2 scheme makes it faster with interpreted languages such as MATLAB.

For the TNNP model, the SBDF2 method outclasses the Strang splitting methods in terms of efficiency. The third-order DC3 method was the most efficient only in the case of extremely accurate solutions. In practice, simulations do not require this much accuracy as the balance between modelling and numerical error allows for a lower level of numerical accuracy.

If the goal is not to have extremely high accuracy but rather smaller computational times, the stability of the methods is the important factor. The stiffness of the more complex models call for more stable methods, such as Rush-Larsen type methods. As we have seen in Chapter 3, these are almost unconditionally stable. However, in Chapter 4, we saw that they are less accurate than other semi-implicit methods such as SBDF2 or Strang splitting methods. For the stiffer models such as TNNP, a lower level of accuracy is not available with the less stable SBDF2 and Strang splitting methods, because even at the largest possible time-step necessary for stability, the errors are extremely low.

Future Work

As we have seen in Chapter 3, the stability analysis that we did for the 1D monodomain model for the semi-implicit methods corresponds to the numerical results obtained for the 2D case. Although it is expected the results also extend to the 3D case, numerical tests could be performed to verify this.

The monodomain model is usually realistic enough for many practical applications. However, in some cases the use of the bidomain model is needed. Results obtained for the monodomain model usually extend to the bidomain model, but this is not generally easily shown. One could try to extend the theoretical analysis on the stability of the time-stepping methods to the bidomain model. This would then need to be supported by additional numerical tests.

The computational cost for realistic simulations in cardiac electrophysiology can be enormous. Most of our simulations were done in the 1D case, and in the 2D case

we had a very simple square domain. Solving the mono- or bi- domain models on realistic heart geometries requires the use of more advanced techniques such as the use of preconditionners, mesh adaptation, parallel computing, etc. The impact of the time-stepping method on the required linear system solver has not been thoroughly researched. The implementation of these techniques is not usually an easy task, therefore researchers tend to favor simpler time-stepping methods, often of first-order accuracy in time, if they lead to simpler or no solution of linear systems (e.g. with FE).

Although we covered many of the popular time-stepping methods used in cardiac electrophysiology, there are several more to be tested. There exist other variations of the Rush-Larsen method, different choices for the methods used in the substeps of the Strang splitting method, and other semi-implicit methods. We hope that our methodology for analyzing the stability of time-stepping schemes can be easily extended to most of these methods.

Appendix A

ten Tuscher-Noble-Noble-Panfilov model

Here we give the details for the TNNP model for epicardial cells [35]. This model has twelve gating variables, denoted m, h, j, d, f, f_{Ca}, r, s, x_s, x_{r1}, x_{r2} and g, and four ionic concentrations: $\mathrm{Ca_i}$, $\mathrm{Na_i}$, $\mathrm{K_i}$ and $\mathrm{Ca_{sr}}$. The source term is given by

$$I_{\mathrm{ion}} = I_{\mathrm{Na}} + I_{\mathrm{K1}} + I_{\mathrm{to}} + I_{\mathrm{Kr}} + I_{\mathrm{Ks}} + I_{\mathrm{CaL}} + I_{\mathrm{NaCa}} + I_{\mathrm{NaK}} + I_{\mathrm{pCa}} + I_{\mathrm{pK}} + I_{\mathrm{bCa}} + I_{\mathrm{bNa}}.$$

Here are the rate constants, steady-state values and time constants for the different gating variables:

$$m_\infty = \frac{1}{(1+e^{(-56.86-u)/9.03})^2} \qquad \alpha_m = \frac{1}{1+e^{(-60-u)/5}}$$

$$\beta_m = \frac{0.1}{1+e^{(u+35)/5}} + \frac{0.1}{1+e^{(u-50)/200}} \qquad \tau_m = \alpha_m \beta_m$$

$$h_\infty = \frac{1}{(1+e^{(u+71.55)/7.43})^2} \qquad \alpha_h = \begin{cases} 0 & \text{if } u \geq -40 \\ 0.057e^{-(u+80)/6.8} & \text{otherwise} \end{cases}$$

$$\beta_h = \begin{cases} \dfrac{0.77}{0.13(1+e^{-(u+10.66)/11.1})} & \text{if } u \geq -40 \\ 2.7e^{0.079u} + 3.1\times 10^5 e^{0.3485u} & \text{otherwise} \end{cases} \qquad \tau_h = \frac{1}{\alpha_h + \beta_h}$$

$$j_\infty = \frac{1}{(1+e^{(u+71.55)/7.43})^2} \qquad \beta_j = \begin{cases} \dfrac{0.6e^{0.057u}}{1+e^{-0.1(u+10.66)}} & \text{if } u \geq -40 \\ \dfrac{0.02424e^{-0.01052u}}{1+e^{-0.1378(u+40.14)}} & \text{otherwise} \end{cases}$$

$$\alpha_j = \begin{cases} 0 & \text{if } u \geq -40 \\ \dfrac{\left(-2.5428\times10^4 e^{0.2444u} - 6.948\times10^{-6}e^{-0.04391u}\right)(u+37.78)}{1+e^{0.311(u+79.23)}} & \text{otherwise} \end{cases}$$

$$\tau_j = \frac{1}{\alpha_j + \beta_j} \qquad d_\infty = \frac{1}{1+e^{(-5-u)/7.5}}$$

$$\alpha_d = \frac{1.4}{1+e^{(-35-u)/13}} + 0.25 \qquad \beta_d = \frac{1.4}{1+e^{(u+5)/5}}$$

$$\gamma_d = \frac{1}{1+e^{(50-u)/20}} \qquad \tau_d = \alpha_d\beta_d + \gamma_d$$

$$f_\infty = \frac{1}{1+e^{(v+20)/7}} \qquad \tau_f = 1125e^{-(u+27)^2/240} + \frac{1}{1+e^{(25-u)/10}} + 80$$

$$\alpha_{f_{\mathrm{Ca}}} = \frac{1}{1+(\mathrm{Ca_i}/0.000325)^8} \qquad \beta_{f_{\mathrm{Ca}}} = \frac{0.1}{1+e^{(\mathrm{Ca_i}-0.0005)/0.0001}}$$

$$\gamma_{f_{\mathrm{Ca}}} = \frac{0.2}{1+e^{(\mathrm{Ca_i}-0.00075)/0.0008}} \qquad f_{\mathrm{Ca}\infty} = \frac{\alpha_{f_{\mathrm{Ca}}} + \beta_{f_{\mathrm{Ca}}} + \gamma_{f_{\mathrm{Ca}}} + 0.23}{1.46}$$

$$\tau_{f_{\mathrm{Ca}}} = 2\,\mathrm{ms} \qquad \frac{df_{\mathrm{Ca}}}{dt} = k\frac{f_{\mathrm{Ca}\infty} - f_{\mathrm{Ca}}}{\tau_{f_{\mathrm{Ca}}}}$$

$$k = \begin{cases} 0 & \text{if } f_{\mathrm{Ca}\infty} > f_{\mathrm{Ca}} \quad \text{and } u > -60\,\mathrm{mV} \\ 1 & \text{otherwise} \end{cases} \qquad r_\infty = \frac{1}{1+e^{(20-u)/6}}$$

$$\tau_r = 9.5e^{-(u+40)^2/1800} + 0.8 \qquad s_\infty = \frac{1}{1+e^{(u+20)/5}}$$

$$\tau_s = 85e^{-(u+45)^2/320} + \frac{5}{1+e^{(u-20)/5}} + 3 \qquad x_{s\infty} = \frac{1}{1+e^{(-5-u)/14}}$$

$$\alpha_{x_s} = \frac{1,100}{\sqrt{1+e^{(-10-u)/6}}} \qquad \beta_{x_s} = \frac{1}{1+e^{(u-60)/20}}$$

$$\tau_{x_s} = \alpha_{x_s}\beta_{x_s} \qquad x_{r1\infty} = \frac{1}{1+e^{(-26-u)/7}}$$

$$\alpha_{x_{r1}} = \frac{450}{1+e^{(-45-u)/10}} \qquad \beta_{x_{r1}} = \frac{6}{1+e^{(u+30)/11.5}}$$

$$\tau_{x_{r1}} = \alpha_{x_{r1}}\beta_{x_{r1}} \qquad x_{r2\infty} = \frac{1}{1+e^{(v+88)/24}}$$

$$\alpha_{x_{r2}} = \frac{3}{1+e^{(-60-u)/20}} \qquad \beta_{x_{r2}} = \frac{1.12}{1+e^{(u-60)/20}}$$

$$\tau_{x_{r2}} = \alpha_{x_{r2}}\beta_{x_{r2}} \qquad g_\infty = \begin{cases} \dfrac{1}{1+\mathrm{Ca_i}^6/0.00035^6} & \text{if } \mathrm{Ca_i} \le 0.00035 \\ \dfrac{1}{1+\mathrm{Ca_i}^{16}/0.00035^{16}} & \text{otherwise} \end{cases}$$

$$\tau_g = 2\,\mathrm{ms} \qquad \frac{dg}{dt} = k\frac{g_\infty - g}{\tau_g}$$

$$k = \begin{cases} 0 & \text{if } g_\infty > g \quad \text{and } u > -60\,\mathrm{mV} \\ 1 & \text{otherwise} \end{cases}$$

The concentrations are defined as follows:

$$\mathrm{Ca_{ibufc}} = \frac{\mathrm{Ca_i Buf_c}}{\mathrm{Ca_i} + \mathrm{K_{bufc}}},$$

$$\frac{d\mathrm{Ca_{itotal}}}{dt} = -\frac{I_{\mathrm{CaL}} + I_{\mathrm{bCa}} + I_{\mathrm{pCa}} - 2I_{\mathrm{NaCa}}}{2\mathrm{V}_C F} + I_{\mathrm{leak}} - I_{\mathrm{up}} + I_{\mathrm{rel}},$$

where $\mathrm{Ca_{itotal}}$ is the sum of $\mathrm{Ca_{ibufc}}$ and $\mathrm{Ca_i}$.

$$\mathrm{Ca_{srbufsr}} = \frac{\mathrm{Ca}_{sr}\mathrm{Buf_{sr}}}{\mathrm{Ca}_{sr} + \mathrm{K_{bufsr}}},$$

$$\frac{d\mathrm{Ca_{srtotal}}}{dt} = \frac{\mathrm{V}_C}{\mathrm{V_{SR}}}(-I_{\mathrm{leak}} + I_{\mathrm{up}} - I_{\mathrm{rel}})$$

where $\mathrm{Ca_{srtotal}}$ is the sum of $\mathrm{Ca_{srbufsr}}$ and $\mathrm{Ca_{sr}}$.

$$\frac{d\mathrm{Na_i}}{dt} = -\frac{I_{\mathrm{Na}} + I_{\mathrm{bNa}} + 3I_{\mathrm{NaK}} + 3I_{\mathrm{NaCa}}}{\mathrm{V}_C F},$$

$$\frac{d\mathrm{K_i}}{dt} = -\frac{I_{\mathrm{K1}} + I_{\mathrm{to}} + I_{\mathrm{Kr}} + I_{\mathrm{Ks}} - 2I_{\mathrm{NaK}} + I_{\mathrm{pK}} + I_{\mathrm{stim}} - I_{\mathrm{ax}}}{\mathrm{V}_C F}.$$

Note that one can retrieve $\frac{d\mathrm{Ca_i}}{dt}$ and $\frac{d\mathrm{Ca_{sr}}}{dt}$ using the chain-rule. The ionic currents are defined using the following:

$$E_X = \frac{\mathrm{RT}}{zF}\log\frac{X_\mathrm{o}}{X_\mathrm{i}} \quad \text{for} \quad X = \mathrm{Na}^+,\ \mathrm{K}^+,\ \mathrm{Ca}^{2+}$$

$$E_{\mathrm{K_s}} = \frac{\mathrm{RT}}{F}\log\frac{\mathrm{K_o} + p_{\mathrm{KNa}}\mathrm{Na_o}}{\mathrm{K_i} + p_{\mathrm{KNa}}\mathrm{Na_i}} \qquad I_{\mathrm{Na}} = G_{\mathrm{Na}} m^3 hj(V - E_{\mathrm{Na}})$$

$$I_{\mathrm{CaL}} = G_{\mathrm{CaL}} df f_{\mathrm{Ca}} 4\frac{VF^2}{\mathrm{RT}}\frac{\mathrm{Ca_i}e^{2VF/\mathrm{RT}} - 0.341\mathrm{Ca_o}}{e^{2VF/\mathrm{RT}} - 1} \qquad I_{\mathrm{to}} = G_{\mathrm{to}} rs(V - E_{\mathrm{K}})$$

$$I_{\mathrm{Ks}} = G_{\mathrm{Ks}} x_s^2 (V - E_{\mathrm{Ks}}) \qquad I_{\mathrm{Kr}} = G_{\mathrm{Kr}}\sqrt{\frac{\mathrm{K_o}}{5.4}} x_{r1} x_{r2}(V - E_{\mathrm{K}})$$

$$I_{\mathrm{K1}} = G_{\mathrm{K1}}\sqrt{\frac{\mathrm{K_o}}{5.4}} x_{\mathrm{K1}\infty}(V - E_{\mathrm{K}}) \qquad \alpha_{\mathrm{K1}} = \frac{0.1}{1 + e^{0.06(u - E_{\mathrm{K}} - 200)}}$$

$$\beta_{\mathrm{K1}} = \frac{3e^{0.0002(u - E_{\mathrm{K}} + 100)} + e^{0.1(u - E_{\mathrm{K}} - 10)}}{1 + e^{-0.5(u - E_{\mathrm{K}})}} \qquad x_{\mathrm{K1}\infty} = \frac{\alpha_{\mathrm{K1}}}{\alpha_{\mathrm{K1}} + \beta_{\mathrm{K1}}}$$

$$I_{\mathrm{NaCa}} = k_{\mathrm{NaCa}}\frac{e^{\gamma VF/\mathrm{RT}}\mathrm{Na_i}^3\mathrm{Ca_o} - e^{(\gamma-1)VF/\mathrm{RT}}\mathrm{Na_o}^3\mathrm{Ca_i}\alpha}{K_{\mathrm{mNai}}^3 + \mathrm{Na_o}^3)(K_{\mathrm{mCa}} + \mathrm{Ca_o})(1 + k_{\mathrm{sat}}e^{(\gamma-1)VF/\mathrm{RT}})}$$

$$I_{\mathrm{NaK}} = P_{\mathrm{NaK}}\frac{\mathrm{K_o}\mathrm{Na_i}}{(\mathrm{K_o} + \mathrm{K_{mK}})(\mathrm{Na_i} + \mathrm{K_{mNa}})(1 + 0.1245e^{-0.1VF/\mathrm{RT}} + 0.0353e^{-VF/\mathrm{RT}})}$$

$$I_{\mathrm{pCa}} = G_{\mathrm{pCa}}\frac{\mathrm{Ca_i}}{\mathrm{K_{pCa}} + \mathrm{Ca_i}} \qquad I_{\mathrm{Kr}} = G_{\mathrm{pK}}\frac{V - E_{\mathrm{K}}}{1 + e^{(25-u)/5.98}}$$

$$I_{\mathrm{pCa}} = G_{\mathrm{bNa}}(V - E_{\mathrm{Na}}) \qquad I_{\mathrm{pCa}} = G_{\mathrm{bCa}}(V - E_{\mathrm{Ca}})$$

$$I_{\mathrm{leak}} = V_{\mathrm{leak}}(\mathrm{Ca_{sr}} - \mathrm{Ca_i}) \qquad I_{\mathrm{up}} = \frac{V_{\mathrm{maxup}}}{1 + K_{\mathrm{up}}^2/\mathrm{Ca_i}^2}$$

$$I_{\mathrm{rel}} = \left(a_{\mathrm{rl}}\frac{\mathrm{Ca_{sr}}^2}{b_{\mathrm{rel}}^2 + \mathrm{Ca_{sr}}^2} + c_{\mathrm{rel}}\right) dg$$

The different values for our simulations of the TNNP model can be found in Table A.1. They have been taken from CellML at `http://models.cellml.org/exposure/c7f7ced1e002d9f0af1b56b15a873736/tentusscher_noble_noble_panfilov_2004_b.cellml/view`.

Table A.1: TNNP model paramters

Parameter	Value	Unit
R	8,314.472	$\mathrm{mJK^{-1}mol^{-1}}$
T	310	K
F	96,485.3415	$\mathrm{A\,s\,mol^{-1}}$
V_C	0.016404	$\mathrm{mm^3}$
$\mathrm{V_{SR}}$	0.001094	$\mathrm{mm^3}$
$\mathrm{Na_o}$	140	mM
$\mathrm{Ca_o}$	2	mM
$\mathrm{K_o}$	5.4	mM
G_{Na}	14.838	$\mathrm{S\,mF^{-1}}$
G_{K1}	5.405	$\mathrm{S\,mF^{-1}}$
G_{to}	0.294	$\mathrm{S\,mF^{-1}}$
G_{Kr}	0.096	$\mathrm{S\,mF^{-1}}$
G_{Ks}	0.245	$\mathrm{S\,mF^{-1}}$
P_{KNa}	0.03	-
G_{Kr}	1.75×10^{-4}	$\mathrm{cm^3\,\mu F\,s^{-1}}$
G_{CaL}	1000	$\mathrm{A\,F^{-1}}$
γ	0.35	-
K_{mCa}	1.38	mM
K_{mNai}	87.5	mM

Parameter	Value	Unit
k_{sat}	0.1	-
α	2.5	-
P_{NaK}	1.362	$\mathrm{A\,F^{-1}}$
K_{mK}	1	mM
K_{mNa}	40	mM
G_{pK}	0.0146	$\mathrm{S\,mF^{-1}}$
G_{pCa}	0.825	$\mathrm{S\,mF^{-1}}$
K_{pCa}	0.0005	mM
G_{bNa}	2.9×10^{-4}	$\mathrm{S\,mF^{-1}}$
G_{bCa}	5.92×10^{-4}	$\mathrm{S\,mF^{-1}}$
V_{maxup}	5.92×10^{-4}	$\mathrm{mM\,ms^{-1}}$
K_{up}	2.5×10^{-4}	mM
a_{rel}	0.016464	mM
b_{rel}	0.25	mM
c_{rel}	0.008232	$\mathrm{mM\,ms^{-1}}$
V_{leak}	8×10^{-5}	$\mathrm{ms^{-1}}$
$\mathrm{Buf_c}$	0.15	mM
$\mathrm{K_{bufc}}$	0.001	mM
$\mathrm{Buf_{sr}}$	10	mM
$\mathrm{K_{bufsr}}$	0.3	mM

Appendix B

Model Parameters and Resting Values

Here we list the parameters used for the monodomain model simulations and the resting values used as initial conditions for each ionic model. The same parameters have been used for the BR and TNNP models, shown in Table B.1 for the 1D case and in Table B.2 for the 2D case. A nondimensionalized version of the MS model [27] has been used and the parameters are shown in Table B.3. The resting values for the BR and TNNP models are specified in Tables B.4 and B.5, respectively. For the nondimensionalized MS model, we use $u = 0$ and $v = 0.99$ for the resting values.

Table B.1: Parameters for the 1D monodomain model

Parameter	Value	Unit
χ	500	cm^{-1}
C_m	1	$\mu\mathrm{F\,cm}^{-2}$
σ_i	17.41	$\mathrm{mS\,cm}^{-1}$
σ_e	39.06	$\mathrm{mS\,cm}^{-1}$

Table B.2: Parameters for the 2D monodomain model

Parameter	Value	Unit
χ	1000	cm^{-1}
C_m	1	$\mu\mathrm{F\,cm}^{-2}$
σ_i^l	1.741	$\mathrm{mS\,cm}^{-1}$
σ_e^l	3.906	$\mathrm{mS\,cm}^{-1}$
σ_i^t	0.1934	$\mathrm{mS\,cm}^{-1}$
σ_e^t	1.970	$\mathrm{mS\,cm}^{-1}$

Table B.3: Parameters for the nondimensionalization of the 1D MS monodomain model

Parameter	Value	Unit
χ	2×10^5	m^{-1}
C_m	0.01	$\mathrm{F\,m}^{-2}$
σ_i	1.741	$\mathrm{mS\,m}^{-1}$
σ_e	3.906	$\mathrm{mS\,m}^{-1}$
L	0.0001	m
T	0.001	s
V_m	0.125	V

Table B.4: Resting Values for the BR model

Variable	Value	Unit
u	$-0.845\,737\,561\,2 \times 10^{2}$	mV
x	$0.562\,865\,055\,5 \times 10^{-2}$	-
m	$0.109\,819\,687\,2 \times 10^{-1}$	-
h	$0.987\,721\,175\,5$	-
j	$0.974\,838\,139\,0$	-
d	$0.297\,072\,466\,3 \times 10^{-2}$	-
f	$0.999\,981\,333\,9$	-
$[\mathrm{Ca}]_{\mathrm{i}}$	$0.178\,200\,721\,6 \times 10^{-6}$	mM

Table B.5: Resting Values for the TNNP model

Variable	Value	Unit
u	-86.424	mV
m	0.133×10^{-2}	-
h	0.776	-
j	0.776	-
d	0.193×10^{-4}	-
f	1.00	-
f_{Ca}	1.00	-
r	0.00	-
s	1.00	-
x_s	0.297×10^{-2}	-
x_{r1}	0.00	-
x_{r2}	0.485	-
g	1.00	-
$\mathrm{Na_i}$	11.212	mM
$\mathrm{K_i}$	137.614	mM
$\mathrm{Ca_i}$	0.445×10^{-4}	mM
$\mathrm{Ca_{sr}}$	0.513	mM

Bibliography

[1] U. M. Ascher, S. J. Ruuth, and B. T. R. Wetton. Implicit-explicit methods for time-dependent partial differential equations. *SIAM J. Numer. Anal.*, 32(3):797–823, 1995.

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

[3] N.F. Britton. *Essential Mathematical Biology.* Springer Science & Business Media, 2003.

[4] J. W. Cain. Taking math to heart: mathematical challenges in cardiac electrophysiology. *Notice of the AMS*, 58(4):542–549, 2011.

[5] Statistics Canada. Mortality, summary list of causes, 2008.

[6] P.G. Ciarlet. *The Finite Element Method for Elliptic Problems*, volume 4 of *Stud. Math. Appl.* North Holland, Amsterdam, 1978.

[7] R.H. Clayton, O. Bernus, E.M. Cherry, H. Dierckx, F.H. Fenton, L. Mirabella, A.V. Panfilov, F.B. Sachse, G. Seemann, and H. Zhang. Models of cardiac tissue electrophysiology: progress, challenges and open questions. *Progress in biophysics and molecular biology*, 104(1):22–48, 2011.

[8] M. Ethier and Y. Bourgault. Semi-implicit time-discretization schemes for the bidomain model. *SIAM J. Numer. Anal.*, 46(5):2443–2468, 2008.

[9] S.K. Godunov. A difference method for numerical calculation of discontinuous solutions of the equations of hydrodynamics. *Mat. Sbornik.*, 89(3):271–306, 1959.

[10] J.-L. Guermond and P.D. Minev. High-order artificial compressibility for the navier-stokes equations, Part I: Theory. *preprint*, 2014.

[11] E. Hairer, S.P. Nørsett, and G. Wanner. *Solving Ordinary Differential Equations I: Nonstiff Problems*, volume 8 of *Springer Series in Comp. Math.* Springer, 1993.

[12] E. Hairer and G. Wanner. *Solving Ordinary Differential Equations II: Stiff Problems and Differential-Algebraic Problems*, volume 14 of *Springer Series in Comp. Math.* Springer, 1996.

[13] M. Hanslien, R. Artebrant, A. Tveito, G. T. Lines, and X. Cai. Stability of two time-integrators for the Aliev-Panfilov system. *Int. J. Numer. Anal. and Model.*, 8(3):427–442, 2011.

[14] M. Hanslien, K.H. Kenneth, and A. Tveito. On a finite difference scheme for a Beeler-Reuter based model of cardiac electrical activity. *Int. J. Numer. Anal. and Model.*, 3(4):395–412, 2006.

[15] A.L. Hodgkin and A.F. Huxley. A quantitative description of membrane current and its applications to conduction and excitation in nerves. *J. Physiol.*, 117:500–544, 1952.

[16] W. Han K. Atkinson and D.E. Stewart. *Numerical Solution of Ordinary Differential Equations.* Wiley, Hoboken, 2009.

[17] J. Keener and J. Sneyd. *Mathematical Physiology.* Springer, 2004.

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

[19] W. Kress and B. Gustafsson. Deferred correction methods for initial boundary value problems. *J. Sci. Comp.*, 17(1-4):241–251, 2002.

[20] J.D. Lambert. *Numerical methods for ordinary differential systems: The initial value problem*. Wiley, Chichester, 1991.

[21] R.J. LeVeque. *Finite difference methods for ordinary and partial differential equations: steady-state and time-dependent problems*, volume 98. SIAM, 2007.

[22] M.E. Marsh, S.Z. Torabi, and R.J. Spiteri. The secrets to the success of the Rush-Larsen method and its generalizations. *Biomedical Engineering, IEEE Transactions on*, 59(9):2506–2515, 2012.

[23] C. Mitchell and D. Schaeffer. A two-current model for the dynamics of cardiac membrane. *Bull. Math. Bio.*, 65:767–793, 2003.

[24] M. Perego and A. Veneziani. An efficient generalization of the Rush-Larsen method for solving electro-physiology membrane equations. *Electronic Transactions on Numerical Analysis*, 35:234–256, 2009.

[25] S. Puwal and B.J. Roth. Forward euler stability of the bidomain model of cardiac tissue. *Biomedical Engineering, IEEE Transactions on*, 54(5):951, 2007.

[26] A. Quarteroni and A. Valli. *Numerical Approximation of Partial Differential Equations*, volume 23 of *Springer Series in Comp. Math.* Springer, 2008.

[27] M. Rioux and Y. Bourgault. A predictive method allowing the use of a single ionic model in numerical cardiac electrophysiology. *ESAIM: Mathematical Modelling and Numerical Analysis*, 47:987—-1016, 2013.

[28] S. Rush and H. Larsen. A practical algorithm for solving dynamic membrane equations. *Biomedical Engineering, IEEE Transactions on*, 25(4):389–392, 1978.

[29] H.J. Schroll, G.T. Lines, and A. Tveito. On the accuracy of operator splitting for the monodomain model of electrophysiology. *International Journal of Computer Mathematics*, 84(6):871–885, 2007.

[30] R.J. Spiteri and R. Dean. Stiffness analysis of cardiac electrophysiological models. *Ann. Biomed. Eng.*, 38:3592–3604, 2010.

[31] R.J. Spiteri and R.C. Dean. On the performance of an implicit–explicit runge–kutta method in models of cardiac electrical activity. *Biomedical Engineering, IEEE Transactions on*, 55(5):1488–1495, 2008.

[32] G. Strang. On the construction and comparison of difference schemes. *SIAM J. Numer. Anal.*, 5:506–517, 1968.

[33] J. Sundnes, R. Artebrant, O. Skavhaug, and A. Tveito. A second-order algorithm for solving dynamic cell membrane equations. *Biomedical Engineering, IEEE Transactions on*, 56(10):2546–2548, 2009.

[34] J. Sundnes, G.T. Lines, B.F. Nielsen X. Cai, K.-A. Mardal, and A. Tveito. *Computing the Electrical Activity in the Heart*, volume 1 of *Monogr. Comput. Sci. Eng.* Springer, 2006.

[35] K.H.W. ten Tusscher, D. Noble, P.J. Noble, and A. Panfilov. A model for human ventricular tissue. *Am. J. Physiol. Heart. Circ. Physiol.*, 286:H1573–H1589, 2004.

[36] S. Torabi Z., M.E. Marsh, J. Sundnes, and R.J. Spiteri. Stable time integration suppresses unphysical oscillations in the bidomain model. *Computational Physics*, 2:40, 2014.

[37] J.A. Trangenstein and C. Kim. Operator splitting and adaptive mesh refinement for the Luo–Rudy I model. *Journal of Computational Physics*, 196(2):645–679, 2004.

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

[39] R.L. Winslow, D.F. Scollan, A. Holmes, C.K. Yung, J. Zhang, and M.S. Jafri. Electrophysiological modeling of cardiac ventricular function: from cell to organ. *Annual review of biomedical engineering*, 2:119, 2000.