UiO : Matematisk institutt
Det matematisk-naturvitenskapelige fakultet

Explicit Time Stepping Schemes for the Bidomain Model

Christian Bjørland

Masteroppgave, våren 2018

Denne masteroppgaven er levert inn under masterprogrammet *Anvendt matematikk og mekanikk*, studieretning *Computational Science*, ved Matematisk institutt, Universitetet i Oslo. Oppgaven er normert til 30 studiepoeng.

Forsiden viser et utsnitt av rotsystemet til den eksepsjonelle liegruppen E_8, projisert ned i planet. Liegrupper ble oppfunnet av den norske matematikeren Sophus Lie (1842–1899) for å uttrykke symmetriene til differensiallikninger og spiller i dag en sentral rolle i flere deler av matematikken.

Abstract

The bidomain and monodomain models describing cardiac electrophysiology are computationally demanding to solve. Employing efficient numerical methods is therefore important for the practical use of these models. In this thesis we have explored the efficiency of numerical methods based on finite elements in space and explicit and semi-implicit finite differences in time. The explicit scheme which solves the bidomain equations uses a fixed number of Jacobi-iterations to solve the stationary, elliptic part of the model. The results show that an explicit scheme based on Jacobi iterations can give comparable and, in some cases, better computational efficiency than semi-implicit schemes based on operator splitting.

Acknowledgements

I want to thank my advisors Joakim Sundnes and Kent-Andre Mardal for their help and inspiration during my work on this thesis. Especially thanks to Joakim for always being available for discussions. I would also like to thank my family, my friends and my dear Veronica for their support.

Contents

Chapter 1

Introduction

Many heart problems are linked to disturbances in the electrical activity. An increased understanding of the electrical activity in the heart could therefore improve our ability to diagnose and treat heart problems.

At the organ-level, the electrical activity is the result of how billions of small-scale processes in the cells interact. We have detailed knowledge of these small-scale processes, but our knowledge of how the processes interact is more limited. Mathematical modelling and computer simulations is an approach to gain insight into this area. By formulating quantitative models of small-scale processes and combining them, we can achieve a model at the organ-level.

The bidomain model, first introduced by Tung [11], can be considered as the benchmark model for studying the flow of current in the heart. The model consists of two coupled partial differential equations, which we will describe in more detail in section 2. Unfortunately, it is in general not possible to solve the bidomain equations analytically. Therefore, we will solve the bidomain model on a computer using numerical methods to obtain an approximate solution.

Realistic models of the heart will typically involve spatial grids with 40-50 million computational nodes. Such a high resolution is computationally demanding and since changes in electrical current in the model happen in small time in-

tervals, we need to use small time steps, which increases the computational complexity further. The computational burden is therefore an important bottleneck for physically realistic simulations based on the bidomain model. By using a non-physiological assumption, we can simplify the bidomain model to the monodomain model. Previous research indicates that the CPU requirements can be reduced by a factor of ten when using the monodomain model [10]. Although computationally less demanding to solve, the monodomain model comes at the cost of being physiologically less accurate. An alternative way of reducing the computational burden is to implement efficient numerical schemes. The latter approach has the advantage of reducing the computational burden without sacrificing physiological accuracy. Implementing efficient numerical schemes is therefore an important avenue for research.

Explicit difference methods in time and space have proven to be popular for solving the monodomain model, due to their simplicity and efficiency. However, finite differences are not suitable for more complex geometries, and they cannot be applied to the bidomain equations due to the stationary, elliptic part of the model. Previous work [9], [12], [2] have preferred semi-implicit schemes for solving the bidomain model, since these methods enable us to split complex problems into smaller and simpler parts.

The purpose of this thesis is to implement and analyze numerical schemes for solving the monodomain and bidomain models. Throughout this thesis we use a "method of lines" approach, i.e. we first apply the finite element method for a spatial discretization of the models, to obtain continuous time-dependent systems of equations. These systems will in turn be discretized in time using different numerical methods. We will implement semi-implicit schemes which use operator splitting methods for time discretization, as well as explicit schemes which apply forward differences for time discretization. However, as mentioned above, the bidomain equations have a stationary, elliptic part, for which the forward differences used in the explicit scheme are not applicable. We will therefore use a fixed number of Jacobi iterations to solve this part of the bidomain equations. The implementation and analysis of an explicit scheme based on Jacobi

iterations is an important contribution of this thesis.

After implementing the numerical schemes for both the monodomain and bidomain models, we explore how the explicit schemes compares to the semi-implicit schemes in terms of efficiency and accuracy. We will also investigate how the number of Jacobi iterations for solving the elliptic part of the bidomain equations affects efficiency and accuracy.

The thesis is organized as follows. Chapter 2 introduces the Jacobi iteratiomonodomain and bidomain models. In the same chapter we will also look at numerical methods relevant for solving these models. Chapter 3 derive numerical schemes for solving the monodomain and bidomain models. In chapter 4 we will explore how the explicit schemes compares to the semi-implicit schemes in terms of efficiency and accuracy, while Chapter 5 summarizes the thesis and looks at potential extensions and limitations of our analysis.

Chapter 2

Mathematical Models

We divide the heart tissue into two separate domains: the intracellular and the extracellular. The cell membrane separates these domains and acts as an electric insulator, which allows for a potential difference between the two domains. This potential difference is called the transmembrane potential. Even though the resistance across the membrane is high, it is possible for ions to pass through specific channels in the membrane. The ionic current across the membrane will be captured by the term I_{ion} in the equations below.

The large number of cells in the heart makes it computationally difficult to model each cell as a separate unit. To achieve a model which is computationally feasible and physiological accurate, the bidomain models is based on volume-averaging techniques. Such techniques involve viewing a quantity at a point as an average of that quantity over a small neighborhood around that point. A consequence is that each point in the heart is in both the intra- and extracelluar domain.

Using a quasi-static condition, we have the following current densities in each domain

$$J_i = -M_i \nabla u_i, \tag{2.1}$$

$$J_e = -M_e \nabla u_e, \tag{2.2}$$

where J_i and J_e are the current densities in the domains, M_i and M_e are the conductivities in the domains. u_i is the intracellular potential and u_e is the extracellular potential. Hence, the transmembrane potential v is defined as $v = u_i$ - u_e. We also assume that following conservation equations are satisfied

$$-\nabla \cdot J_i = \frac{\partial q_i}{\partial t} + \kappa I_{ion}, \tag{2.3}$$

$$-\nabla \cdot J_e = \frac{\partial q_e}{\partial t} - \kappa I_{ion}, \tag{2.4}$$

where q_i is the intracellular and q_e is the exctracellular charge. I_{ion} is the ionic current across the membrane. From (2.1) - (2.4) and some additional assumptions [8], we can derive the bidomain equations

$$\frac{\partial s}{\partial t} = f(s, v, t), \tag{2.5}$$

$$\frac{\partial v}{\partial t} + I_{ion}(s, v, t) = \nabla \cdot (M_i \nabla v) + \nabla \cdot (M_i \nabla u_e), \tag{2.6}$$

$$\nabla \cdot (M_i \nabla v) + \nabla \cdot ((M_i + M_e)\nabla u_e) = 0, \tag{2.7}$$

where (2.5) is a system of ODEs that describes the electrophysiological activity in the heart cells. Note that we have introduced the scaled conductivities $M_i = M_i^*/(C_m\chi)$ $M_e = M_*e/(C_m\chi)$, and the scaled ionic current $I_{ion} = I_{ion}^*/C_m$. M_i^*, M_e^* and I_{ion}^* are, respectively, the standard bidomain conductivities and ionic current.

We need boundary conditions for u_e and v to solve the model. For the present study we assume that the heart is surrounded by an insulating material, which we express as

$$n \cdot J_i = 0, \tag{2.8}$$

$$n \cdot J_e = 0, \tag{2.9}$$

Using (2.1) - (2.2) and (2.8) - (2.9), we can eliminate u_i and get

$$n \cdot (M_i \nabla v + M_i \nabla u_e) = 0 \tag{2.10}$$

$$n \cdot (M_e \nabla u_e) = 0 \tag{2.11}$$

The bidomain model can be simplified by assuming equal anisotropy rates, which implies $M_i = \lambda M_e$, where λ is a constant scalar. Using this simplifying assumption, one can derive the monodomain equation

$$\frac{\partial v}{\partial t} + I_{ion}(s, v, t) = \nabla \cdot (\sigma \nabla v)\,, \tag{2.12}$$

$$\frac{\partial s}{\partial t} = f(s, v, t), \tag{2.13}$$

with boundary conditions

$$n \cdot (\sigma \nabla v) = 0. \tag{2.14}$$

As for the bidomain model, we scale the ionic current and the conductivity. The monodomain model is easier to analyse and solve numerically than the bidomain model. However, the monodomain model has some caveats. Firstly, measurements of the conductivities indicate that the assumption of equal conductivity rates is wrong [8]. In addition, there are some electrophysiological phenomena which are not captured by the monodomain model [8].

I used Fenics with high-level scripting in Python to solve the monodomain and bidomain equations. Fenics is open source software, and it aims at being "an easy, intuitive, efficient and flexible software for solving partial differential equations (PDEs) using the finite element method" [4, p. 3] .

Chapter 3

Numerical Methods

We have several options for discretization in time and space, including, but not limited to, finite differences and the finite element method. Previous work [7] on numerical methods for the bidomain model has pointed out that the finite element method are more suitable than finite differences for spatial discretization. We will use finite differences for temporal discretization and the finite element method for spatial discretization.

3.1 The Finite Element Method

When using the finite element method, we must choose the geometry of the elements and basis functions for the trial and test spaces. These choices are important determinants of spatial accuracy and CPU-time. In this thesis, we will use 1D and 2D models with P1 elements and Lagrange polynomials as basis functions. In other words, we will use a uniformly partitioned mesh. One can argue that a more complex geometry with basis functions of a higher degree would give higher spatial accuracy and a more accurate representation of the geometry of the heart. But this thesis focuses on differences in accuracy and efficiency between different numerical schemes, rather than having a physiological accurate representation of the heart. In addition, as we will discuss in the next section, the explicit schemes use a "lumped" mass matrix, which reduces the spatial convergence to one. Thus, we have little to gain in accuracy by using quadratic elements for the explicit schemes. Previous research which have

applied the finite element method for spatial discretization have also used linear elements [2].

3.2 Finite Differences

We will use finite differences for temporal discretization. Which finite difference methods we use will have an impact on accuracy, computational complexity and stability. To investigate the accuracy and complexity of these methods, consider the following system of ODEs.

$$\frac{dy}{dt} = f(y,t) \tag{3.1}$$

Let the time domain $[0,T]$ be partitioned into N equal-sized sub-intervals, i.e. $0 = t^0 < t^1 < ... < t^N = T$, $t^i - t^{i-1} = \Delta t \quad \forall i$. Let $y^n = y(t^n)$. The explicit forward Euler method involves using the following approximation for y^{n+1}.

$$y^{n+1} = y^n + \Delta t f(y^n, t^n) \tag{3.2}$$

We can motivate this approximation by using the following Taylor-expansion of y^{n+1} around t^n.

$$y^{n+1} = y^n + \frac{dy^n}{dt}(t^{n+1} - t^n) + \mathcal{O}(\Delta t^2) \tag{3.3}$$

Inserting for (3.1) in (3.3) and re-arranging, we get

$$y^{n+1} - (y^n + \Delta t f(y^n, t^n)) = \mathcal{O}(\Delta t^2) \tag{3.4}$$

Hence, we get a local error of order two for the forward Euler method. Since $N = \Delta t^{-1}$, we get an accumulated error proportional to Δt. Thus, an explicit time-stepping scheme using the forward Euler method is of order one.

The backward Euler method uses the following approximation for y^{n+1}

$$y^{n+1} = y^n + \Delta t f(y^{n+1}, t^{n+1}) \tag{3.5}$$

The backward Euler method can be derived from the Taylor-expansion of y^n around t^{n+1}

$$y^n = y^{n+1} - \Delta t f(y^{n+1}, t^{n+1}) + \mathcal{O}(\Delta t^2) \tag{3.6}$$

From (3.6), we see that the backward Euler method is of order one. (3.6) is in general a system of nonlinear algebraic equations. To summarize, both methods are of order one, and the the forward Euler method is less computational demanding. However, a caveat with using a forward Euler method is its strict stability requirements. [6] derives the following stability criterion for the forward Euler method for an anisotropic bidomain model in 2D when $\Delta x = \Delta y$.

$$\Delta t \leq \kappa C_m \left(\frac{1}{trace(M_i)} + \frac{1}{trace(M_e)} \right) \Delta x^2 \tag{3.7}$$

In our test cases, we will use scalar values for M_i and M_e. We will also use scaled conductivities and scaled ionic current, see section 4.1 for more details on parameter values. We have to keep in mind that the stability criterion above is derived when using finite differences in time and space, whereas we will use finite differences in time and the finite element method for spatial discretization.

Earlier works have consider both implicit methods [5] and explicit methods [1] for temporal discretization. Operator splitting methods is another alternative. The motivation behind using operator splitting methods is to use split complex system of equations to more manageable parts. After splitting the problem to smaller pieces, we can apply efficient numerical methods. When we derive semi-implicit schemes later in this thesis, we will use explicit methods for computing the non-linear part of the problem, an we will use implicit methods for solving the linear part of the problem. Because of the desirable properties of an operator splitting method, several works [9], [12], [2] have used an operator splitting approach to solve the bidomain model.

When we derive the explicit schemes in section 3.3.2 and section 3.4.2 we approximate the mass matrix by a diagonal matrix. This method is referred to as *mass lumping*. Common ways of lumping a mass matrix involve using each

row sum as a diagonal element, or to use numerical integration with quadrature points at the nodes [3, p. 187]. Another way of lumping the mass matrix involves scaling each diagonal element in the mass matrix by the row sum divided by the sum of all the elements in the mass matrix. This latter method was most successfull, and was therefore used for the explicit schemes for the monodomain and bidomain models.

3.3 Numerical Schemes for the Monodomain Model

3.3.1 Spatial Discretization

The monodomain model is given by (2.12) - (2.13) with boundary condition (2.14). We want to discretize this system with a method of lines approach, i.e., we first apply the finite element method for a spatial discretization of the system, to obtain a system of of ODEs in time.

To apply the finite element method, we introduce an appropriate function space V_h, with basis functions $\phi_j, j = 1, \ldots, N$. The unknown field v is then approximated as linear combinations of the basis functions

$$v = \sum_{j=1}^{N} v_j \phi_j, s = \sum_{j=1}^{N} s_j \phi_j, \tag{3.8}$$

where v_j, s_j are time-dependent coefficients and ϕ_j are appropriate (spatial) basis functions. Note that each v_j is a scalar value (since v is a scalar field), while each s_j is a vector with the same number of components as the vector field s. For simplicity, we will consider the case where s is a scalar field, which corresponds to a Fitzhugh-Nagumo type of model. Specifically, we will use the following cubic polynomial for the ionic term

$$I_{ion} = c_1 v(v - a)(1 - v) - c_2 vs \tag{3.9}$$

and the following functional form for f

$$f = b(v - c_3 s) \tag{3.10}$$

To simplify the notation, we introduce the bilinear form

$$a(\eta, \psi) = \int_\Omega \sigma \nabla \eta \cdot \nabla \psi \, dx,$$

A weak form of (2.12)-(2.13) is

$$\frac{d}{dt} \int_\Omega v \, \psi \, dx = -a(v, \psi) - \int_\Omega I_{ion}(s, v) \, \psi \, dx \tag{3.11}$$

$$\frac{d}{dt} \int_\Omega s \, \psi \, dx = \int_\Omega f(s, v, t) \, \psi \, dx \tag{3.12}$$

and is to be satisfied for all choices $\psi \in V$, where V is some suitable function space. By inserting the approximation (3.8) into (3.11)-(3.12) and using the basis functions $\phi_j, j = 1, \ldots, N$ as test functions ψ, we get a non-linear ODE system

$$M \frac{dv}{dt} = -Av - MI_{ion}, \tag{3.13}$$

$$M \frac{ds}{dt} = Mf, \tag{3.14}$$

Here I_{ion}, f are the vectors of nodal values for $I_{ion}(v, s)$ and $f(v, s, t)$, respectively. We have used a standard finite element interpolation to approximate the fields by their nodal values:

$$I_{ion} \approx \sum_{j=1}^{N} I_{ion}^j \phi_j, f \approx \sum_{j=1}^{N} f_j \phi_j. \tag{3.15}$$

The matrices are given by

$$M_{jk} = \int_\Omega \phi_j \, \phi_k \, dx,$$

$$A_{jk} = \int_\Omega \sigma \, \nabla \phi_j \cdot \nabla \phi_k \, dx,$$

(3.13) - (3.14) is an ODE system, for which we can explore different time discretizations.

3.3.2 Explicit Scheme

We use the forward Euler method because of its simplicity. One could argue that we could have used explicit methods with a higher degree of accuracy, such as a Runge-Kutta method. However, since the temporal resolution is governed by the stability requirement in (3.7), we expect that the spatial error will dominate the error for the explicit scheme. If we apply the Forward Euler method to (3.13) - (3.14), we get

$$Mv^{n+1} = -\Delta t A v^n - M I_{ion}^n + M v^n, \tag{3.16}$$

$$s^{n+1} = \Delta t f^n + s^n, \tag{3.17}$$

where v^n is a vector of the nodal values of v at time step n in the numerical scheme. I_{ion}^n, f^n and s^n are defined similarly. We observe from (3.16) that we need to solve a system of equations for each time step. To reap the potential benefits of an explicit method, we can apply *mass lumping*, by approximating $M \approx M_{diag}$, where M_{diag} is a diagonal matrix. A caveat of this approach is that the spatial convergence is reduced from second-order to first-order. Inserting the approximated mass matrix, we get the system to be computed at each time step for the explicit scheme

$$v^{n+1} = -M_{diag}^{-1} \Delta t A v^n - I_{ion}^n + v^n, \tag{3.18}$$

$$s^{n+1} = \Delta t f^n + s^n, \tag{3.19}$$

Since M_{diag}^{-1} is diagonal, (3.18) is less computationally demanding to solve than (3.16).

3.3.3 Operator Splitting Scheme

If we apply Godunov splitting to (3.13) - (3.14), we get the following two systems to solve at each time step of the algorithm

Step 1 Solve

$$M\frac{dv}{dt} = -Av \qquad s(t_n) = s^n, v(t_n) = v^n, \tag{3.20}$$

for $t^n \leq t \leq t^{n+1}$. We denote the solution at t^{n+1} as v_1^n.

Step 2 Solve

$$\frac{dv}{dt} = -I_{ion}, \qquad v(t^n) = v_1^n, s(t^n) = s^n, \tag{3.21}$$

$$\frac{ds}{dt} = f, \tag{3.22}$$

for $t^n \leq t \leq t^{n+1}$. We denote the solution at t^{n+1} as v^{n+1}.

We apply the backward Euler scheme to (3.20) due to its stability properties and simplicity. It can be shown that Godunov splitting gives first-order convergence [8, p. 71-75]. Hence, there are no benefits to using an implicit scheme with higher accuracy than the Backward Euler method for temporal discretization. We apply the Forward Euler method to the systems of ODEs given by (3.21) and (3.22). Using the temporal discretizations outlined above, we get the following operations to perform at each time step for the splitting-algorithm

Step 1 Solve

$$M(v_1^n - v^n) = -\Delta t A v_1^n \tag{3.23}$$

Step 2 Compute

$$v^{n+1} = -\Delta t I_{ion}^n + v_1^n, \tag{3.24}$$

$$s^{n+1} = \Delta t f^n + s^n, \tag{3.25}$$

3.4 Numerical Schemes for the Bidomain Model

3.4.1 Spatial Discretization

The bidomain equations are given by (2.5) - (2.7) with boundary conditions (2.10) - (2.11). In the following, we use a "methods of lines" approach, i.e. we first use the finite element method for spatial discretization, which gives us a system of differential-algebraic equations.

Let V_h be a suitable function space with basis functions $\phi_j, j = 1, \ldots, N$. We make the following ansatz

$$v = \sum_{j=1}^{N} v_j \phi_j, s = \sum_{j=1}^{N} s_j \phi_j, I = \sum_{j=1}^{N} I_j \phi_j \tag{3.26}$$

where v_j, s_j and I_j are time-dependent coefficients and ϕ_j are appropriate (spatial) basis functions. We will use the same functional forms as we used for the monodomain model, which are given by

$$I_{ion} = c_1 v(v-a)(1-v) - c_2 vs \tag{3.27}$$

$$f = b(v - c_3 s) \tag{3.28}$$

For notational convenience, we use the bilinear forms

$$a_I(\eta, \psi) = \int_\Omega M_i \nabla\eta \cdot \nabla\psi \, dx, \tag{3.29}$$

$$a_{I+E}(\eta, \psi) = \int_\Omega (M_i + M_e) \nabla\eta \cdot \nabla\psi \, dx, \tag{3.30}$$

We choose V_h as the test space of the variational problem. Let $\psi \in V_h$ be a test function. A weak form of (2.5) - (2.7) is

$$\frac{d}{dt} \int_\Omega s\,\psi \, dx = \int_\Omega f\,\psi \, dx \tag{3.31}$$

$$-a_I(v, \psi) - a_I(u_e, \psi) = \frac{d}{dt} \int_\Omega v\,\psi \, dx + \int_\Omega I_{ion}(s, v)\,\psi \, dx \tag{3.32}$$

$$-a_I(v, \psi) - a_{I+E}(u_e, \psi) = 0 \tag{3.33}$$

which must be satisfied $\forall \psi \in V$. Inserting the approximations from (3.26) into (3.31)-(3.33) and using the basis functions $\phi_j, j = 1, \ldots, N$ as test functions ψ, we get the following differential-algebraic system of equations

$$\frac{ds}{dt} = f \tag{3.34}$$

$$M\frac{dv}{dt} + MI_{ion} = -A_1(v + u_e) \tag{3.35}$$

$$A_1 v + A_2 u_e = 0 \tag{3.36}$$

where

$$M_{jk} = \int_\Omega \phi_j \, \phi_k \, dx,$$
$$A_{1jk} = \int_\Omega M_i \nabla \phi_j \cdot \nabla \phi_k \, dx,$$
$$A_{2jk} = \int_\Omega (M_i + M_e) \, \nabla \phi_j \cdot \nabla \phi_k \, dx$$

In (3.34) - (3.36), I_{ion}, f, v and s are the vectors of nodal values for $I_{ion}(v, s)$, $f(v, s, t), v$ and s, respectively. We have used a standard finite element interpolation to approximate the fields by their nodal values:

$$I_{ion} \approx \sum_{j=1}^{N} I_{ion}^j \phi_j, f \approx \sum_{j=1}^{N} f_j \phi_j. \tag{3.37}$$

3.4.2 Explicit Scheme

If we use the forward Euler method, we get the following discretized system of equations

$$s^{n+1} = \Delta t f^n + s^n \tag{3.38}$$

$$Mv^{n+1} = -\Delta t A_1(v^n + u_e^n) - MI_{ion}^n + Mv^n \tag{3.39}$$

$$A_1 v^{n+1} + A_2 u_e^{n+1} = 0 \tag{3.40}$$

where v^n is a vector of the nodal values of v at time step n in the numerical scheme. I_{ion}^n, f^n and s^n are defined similarly. To reduce the computational burden when we solve (3.39), we can apply mass lumping

$$s^{n+1} = \Delta t f^n + s^n \tag{3.41}$$

$$v^{n+1} = -M_{diag}^{-1} \Delta t A_1 (v^n + u_e^n) - \Delta t I_{ion}^n + v^n \tag{3.42}$$

$$A_1 v^{n+1} + A_2 u_e^{n+1} = 0 \tag{3.43}$$

where M_{diag} is a diagonal matrix. (3.41) and (3.42) are straightforward to compute. (3.43) is a system of equations which need to be solved at each time step. As previously mentioned, we use the Jacobi method to find an approximate solution to (3.43). One Jacobi iteration involves the following operation

$$u_{e,k}^n = D^{-1}(R u_{e,k-1}^n - A_1 v^n), \tag{3.44}$$

where $u_{e,k}^n$ is the value of u_e^n at the k'th iteration for the Jacobi method. D is a diagonal matrix with the same diagonal as A_1. R is given by the relation $A_2 = D + R$. We can choose the number of iterations for the Jacobi method to be the smallest integer for which

$$||u_{e,k}^n - u_{e,k-1}^n|| < \epsilon \tag{3.45}$$

where $||.||$ is the $L2$-norm and ϵ is a chosen tolerance. Alternatively, we can use a fixed number of Jacobi iterations for each time step. The latter approach has the advantage of being simple to implement and potentially computationally less demanding for a given number of timesteps. An undesirable property of using a fixed number of Jacobi iterations is that we have less control of the error for each timestep, which can be especially problematic when we only use a few iterations per time step. However, this problem can be alleviated making a good initial guess. Since $||u_e^n - u_e^{n-1}||$ decreases in Δt if u_e^n converges towards the true solution[1], the solution from the previous time step will be a good initial guess for the Jacobi method. If we use a fixed number of iterations, which we denote by K, and use the solution at the previous time step as an initial guess, we get the following algorithm for using Jacobi iterations to solve the elliptic part of the bidomain model

[1] u_e^n is a Cauchy-sequence if u_e^n is convergent

$u_{e,1}^n = -D^{-1}(A_1 v^n + R u_e^{n-1})$

For k $= 2, ..., K$

$u_{e,k}^n = -D^{-1}(A_1 v^n + R u_{e,k-1}^n)$

We are faced with a trade-off when choosing the number of Jacobi iterations to solve the elliptic part. On one hand, an increased number of Jacobi iterations for a given resolution in time and space increases the accuracy of the numerical solution. On the other hand, the CPU-time increases in the number of Jacobi iterations. Since our explicit scheme poses a strict stability requirement on the size of the time step, numerical solutions from this scheme typically have a high temporal resolution. Thus, it is not unreasonable to expect that the time steps which ensures stability for the explicit scheme are sufficiently small to ensure a satisfactory spatial accuracy when we only use one or a few Jacobi iterations. Ideally, we can achieve both a high accuracy and a low CPU-time for our explicit scheme by choosing only one or a few Jacobi iterations. This prediction is an important motivation for the analysis done in this thesis.

We can summarize the computations for each time step for an explicit scheme based on K Jacobi iterations in the following algorithm

$s^{n+1} = \Delta t f^n + s^n$

$v^{n+1} = -M_{diag}^{-1} \Delta t A_1 (v^n + u_e^n) - \Delta t I_{ion}^n + v^n$

$u_{e,1}^{n+1} = -D^{-1}(A_1 v^{n+1} + R u_e^n)$

For k $= 2, ..., K$

$u_{e,k}^{n+1} = -D^{-1}(A_1 v^{n+1} + R u_{e,k-1}^{n+1})$

$u_e^{n+1} = u_{e,K}^{n+1}$

3.4.3 Operator Splitting Scheme

If we apply Godunov splitting to (3.34) - (3.36), we get the following two systems to solve at each time step of the splitting-algorithm

Step 1: Solve

$$\frac{dv}{dt} = -I_{ion}, \qquad v(t^n) = v^n, s(t^n) = s^n, \tag{3.46}$$

$$\frac{ds}{dt} = f, \tag{3.47}$$

for $t^n \leq t \leq t^{n+1}$. We denote the solution at t^{n+1} as v_1^n and s^{n+1}.

Step 2: Solve

$$M\frac{dv}{dt} = -A_1(v + u_e) \qquad v(t_n) = v_1^n, u_e(t_n) = u_e^n, \tag{3.48}$$

$$A_2 u_e + A_1 v = 0 \tag{3.49}$$

for $t^n \leq t \leq t^{n+1}$.We denote the solutions at t^{n+1} as v^{n+1} and u_e^{n+1}.

We apply the forward Euler method to the time derivatives in (3.46) and (3.47). We use a Backward Euler method for temporal discretization of (3.48). If we apply the methods outlined above, we get the following discretized version of the splitting-algorithm

Step 1: Compute

$$v_1^n = -\Delta t I_{ion}^n + v^n \tag{3.50}$$

$$s^{n+1} = \Delta t f^n + s^n, \tag{3.51}$$

Step 2: Solve

$$M(v^{n+1} - v_1^n) = -\Delta t A_1(v^{n+1} + u_e^{n+1}) \tag{3.52}$$

$$A_2 u_e^{n+1} + A_1 v^{n+1} = 0 \tag{3.53}$$

Step 1 in the splitting-algorithm involves only explicit formulas, and is therefore straightforward to compute. The linear systems in Step 2 in the splitting-algorithm can be written as the following block structured linear system

$$\begin{bmatrix} M + \Delta t A_1 & \Delta t A_1 \\ A_1 & A_2 \end{bmatrix} \begin{bmatrix} v^{n+1} \\ u_e^{n+1} \end{bmatrix} = \begin{bmatrix} M v_1^n \\ 0 \end{bmatrix} \tag{3.54}$$

which can be solved by direct or iterative methods. I was unfortunately not able to formulate and solve a block-linear system of equations of the form (3.54) in Fenics. Instead, I used a high-level call to assemble and solve the weak formulation with Newtons method for each time step. Assemblig for each time step is not the fastest way to solve the linear parts of the bidomain model for the operator splitting scheme. We must keep this point in mind when we compare the efficiency of the two different schemes for the bidomain model.

Chapter 4

Results

4.1 Description of the Test Cases

The tests for convergence and efficiency are computed on P1 elements, see section 3.1 for the reasoning behind using this geometry. The spatial domain in 1D is the unit interval, i.e.

$$\Omega = \{x \in \mathbb{R} : 0 \leq x \leq 1\} \tag{4.1}$$

with initial condition

$$v = 1 \qquad \text{for } x \leq 0.5 \tag{4.2}$$

$$v = 0 \qquad \text{for } x > 0.5 \tag{4.3}$$

In 2D, the spatial domain is the unit square

$$\Omega = \{\mathbf{x} \in \mathbb{R}^2 : 0 \leq x, y \leq 1\} \tag{4.4}$$

with initial condition

$$v = 1 \qquad \text{for } ||x||^2 \leq 0.5 \tag{4.5}$$

$$v = 0 \qquad \text{for } ||x||^2 > 0.5 \tag{4.6}$$

where $||.||$ is the Euclidian norm. The time-domain $[0, T]$ is partitioned into N equal-sized sub-intervals, i.e. $0 = t_0 < t_1 < ... < t_N = T$, $t_i - t_{i-1} = \Delta t \quad \forall i$.

Table 4.1 and table 4.2 shows the values of the parameters for the monodomain and bidomain model, respectively. The parameters values were chosen to give a reasonable shape of the action potential. For simplicity, we have used scalar values for M_i and M_e both in 1D and in 2D for the bidomain model.

Table 4.1: Parameter values for the monodomain model

Parameters	Values
a	0.1
b	1
c_1	200
c_2	200
c_3	1
σ	0.1

Table 4.2: Parameter values for the bidomain model

Parameters	Values
a	0.1
b	1
c_1	200
c_2	200
c_3	1
M_i	0.1
M_e	0.1

Figure 4.1 and Figure 4.2 show a numerical solution of the transmembrane potential for the monodomain model in 2D at $t = 0.1$. The latter figure is a 3D-plot.

Figure 4.1: Numerical solution of the transmembrane potential for the bidomain model in 2D at $t = 0.1$. The solution is computed with an operator splitting scheme with resolution $\Delta x = 0.01$ and $\Delta t = 0.00025$. The spatial domain is the unit box.

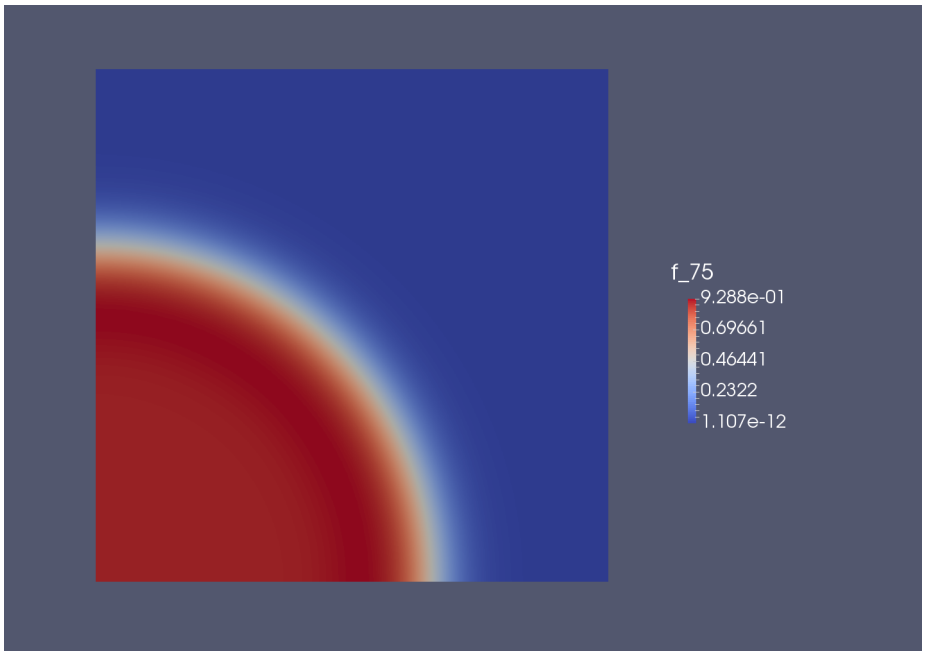


Figure 4.2: Numerical solution of the transmembrane potential for the monodomain model in 2D at $t = 0.1$. The solution is computed with an operator splitting scheme with resolution $\Delta x = 0.01$ and $\Delta t = 0.00025$. The spatial domain is the unit box.

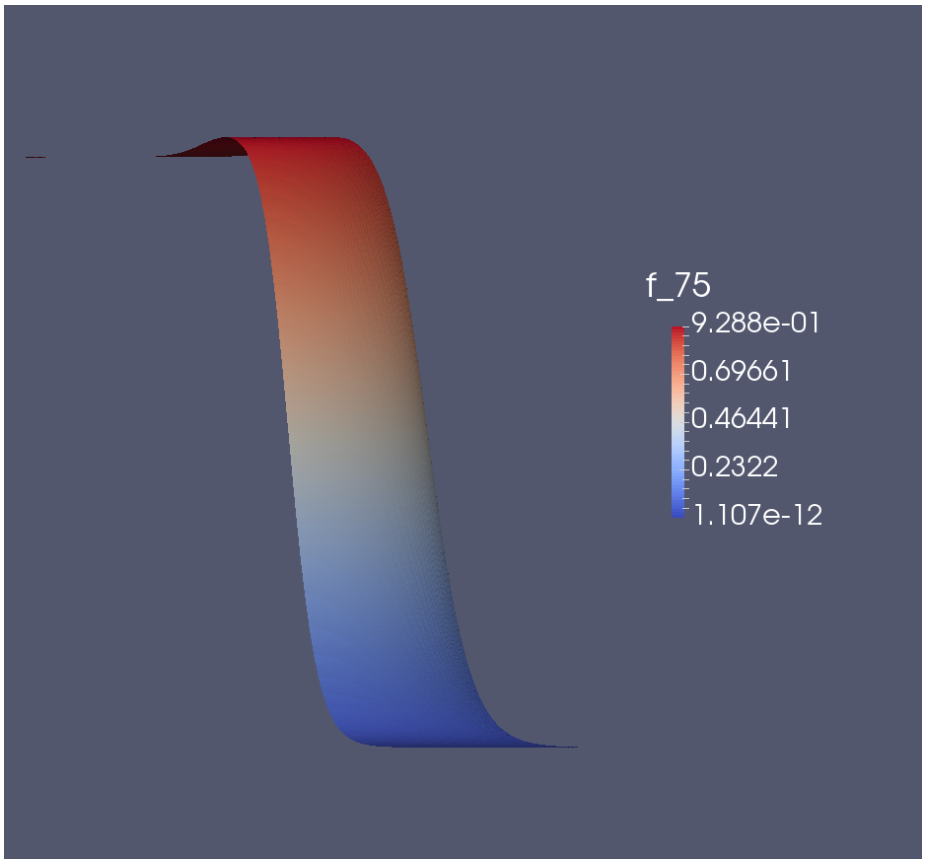


Figure 4.3 and Figure 4.4 show a numerical solution of the transmembrane potential for the monodomain model in 2D at $t = 0.1$. The latter figure is a 3D-plot.

Figure 4.3: Numerical solution of the transmembrane potential for the bidomain model in 2D at $t = 0.1$. The solution is computed with an operator splitting scheme with resolution $\Delta x = 0.01$ and $\Delta t = 0.00025$. The spatial domain is the unit box.

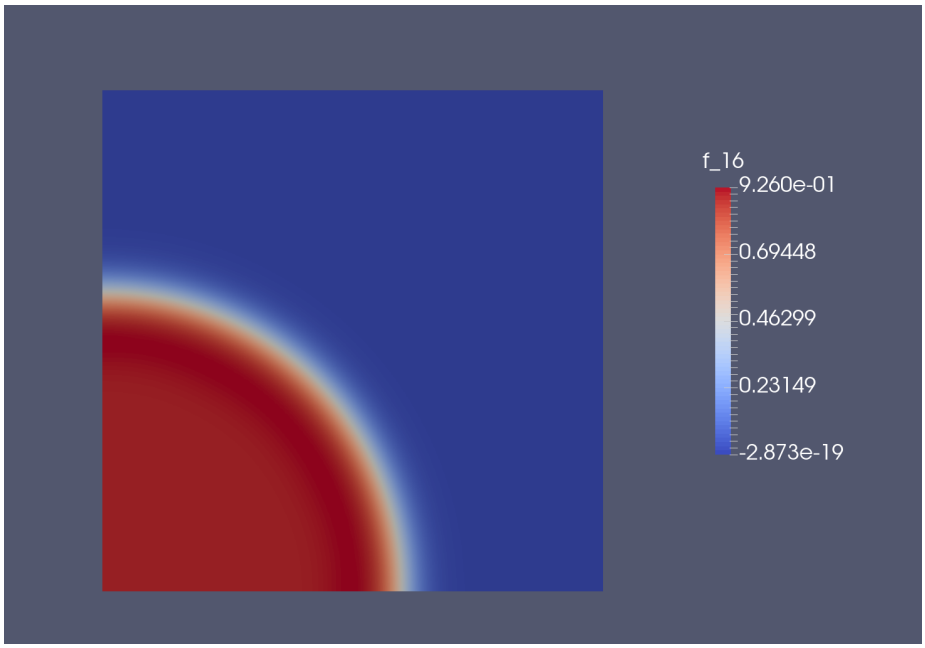
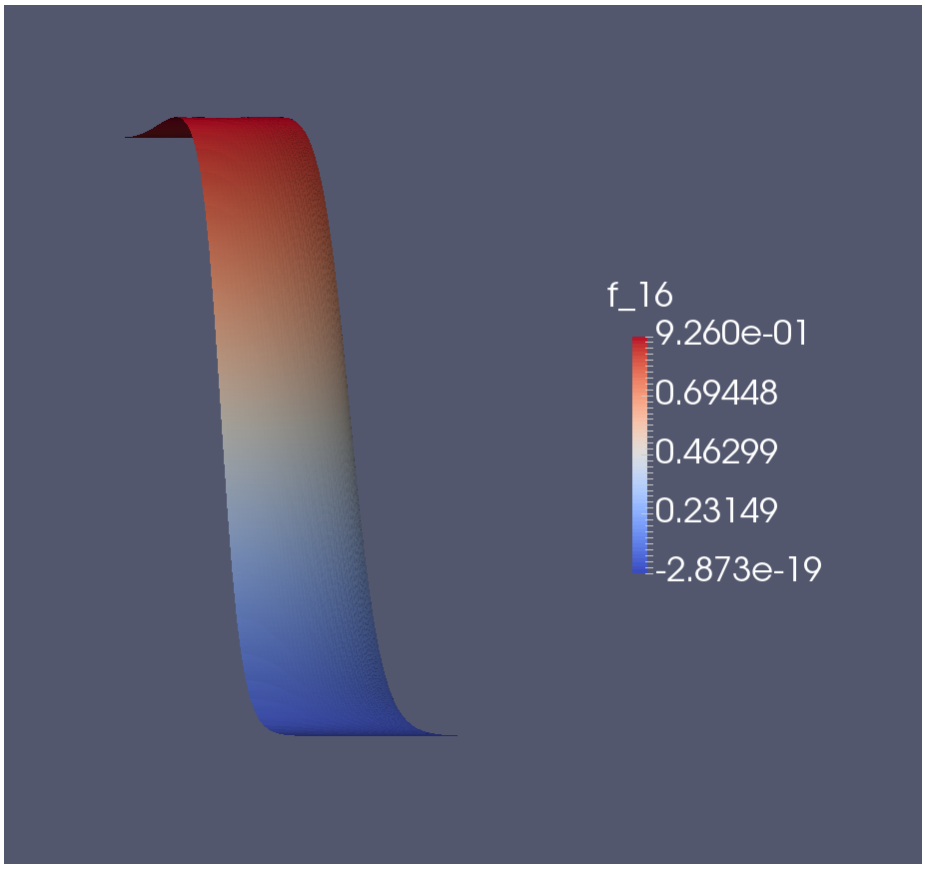


Figure 4.4: Numerical solution of the transmembrane potential for the bidomain model in 2D at $t = 0.1$. The solution is computed with an operator splitting scheme with resolution $\Delta x = 0.01$ and $\Delta t = 0.00025$. The spatial domain is the unit box.



4.2 Error Analysis

We will use a reference solution when we compute the error of our numerical solution. Using a reference solution instead of a analytic solution has some potential problems. Firstly, the computed errors in the numerical experiments is incorrect. However, this problem may be remedied by requiring a high resolution for the reference solution. On the other hand, these resolution requirements may pose a computational challenge. Another problem can be that our numerical method converges to a wrong solution, which is problematic regardless of the

resolution of the reference solution. Based on the discussion above, we should keep in mind potential problems which could arise when using this approach.

We use the discrete $L2$-norm to compute the error. The error of a numerical solution u at time t is

$$E = \left\| u^t - u_r^t \right\| = \left(\sum_{j \in I_x} (u_j^t - u_{r,j}^t)^2 \right)^{1/2} \tag{4.7}$$

where I_x is the index set of the spatial nodes for the numerical solution u, u_j^t is the numerical solution at time t at node j, and u_r^t is the reference solution at time t at node j.

We assume the following relation between the error of the numerical solution and the discretization parameters in time and space

$$E = K_x dx^{r_x} + K_t \Delta t^{r_t} \tag{4.8}$$

where K_x and K_t are constants, r_x is the convergence rate in space and r_t is the convergence rate in time. If we use a common discretization parameter Δ, we get the following estimate of the convergence rate

$$r_i = log(E_{i-1}/E_i)/log(\Delta_{i-1}/\Delta_i) \tag{4.9}$$

where Δ_i is the discretization in experiment i and E_i is the error in experiment i.

4.3 Results for the Monodomain Model

4.3.1 Convergence rates in 1D

Since we "lump" the mass matrix, we expect that the spatial convergence for the explicit scheme is reduced to order one. We also expect first-order conver-

gence in time, since we use the forward Euler method for temporal discretization. Table 4.3 shows convergence results for the explicit scheme applied to the FitzHugh-Nagumo model. The errors are computed at time $t = 0.1$. Since the epxlicit scheme is considerable faster than the semi-impilcit scheme for high resolutions, we used the former scheme to compute the reference solution. The reference solution has resolution $\Delta x = 1/10000$ and $\Delta t = 1/21000000$. We observe that computed convergence rates are close to our theoretical predictions.

Table 4.3: Convergence rates for the explicit scheme for the monodomain model in 1D

Δx	Δt	error	r
0.025000	0.000192	0.0362	
0.012500	0.000096	0.0160	1.1801
0.006250	0.000048	0.0074	1.1047
0.003125	0.000024	0.0035	1.0718

Convergence results for the explicit scheme for the monodomain model in 1D. The reference solution is computed with the explicit scheme and it has resolution $\Delta x = 1/10000$ and $\Delta t = 1/21000000$. The error is computed at $t = 0.1$

We expect first-order convergence in time and second-order convergence in space for our semi-implicit scheme. Table 4.4 shows convergence results for the semi-implicit scheme applied to the FitzHugh-Nagumo model. We use a reference solution computed with the explicit scheme with spatial resolution $\Delta x = 1/10000$ and temporal resolution $\Delta t = 1/21000000$. The reference solution is computed with the explicit scheme. The errors are computed at time $t = 0.1$. We observe that the convergence is approximately second-order in space and first-order in time for the second and third experiments, which is as expected.

Table 4.4: Convergence rates for the operator splitting scheme for the monodomain model in 1D

Δx	Δt	error	r
0.001923	0.002500	0.0198	
0.000962	0.001250	0.0101	0.9724
0.000481	0.000625	0.0051	0.9929
0.000240	0.000313	0.0025	1.0034

Convergence results for the operator splitting scheme for the monodomain model in 1D. The reference solution is computed with the explicit scheme and it has resolution $\Delta x = 1/10000$ and $\Delta t = 1/21000000$. The error is computed at $t = 0.1$.

4.3.2 Efficiency in 1D

Table 4.5 shows the efficiency for both the explicit and semi-implicit scheme applied to the FitzHugh-Nagumo model. We observe that both schemes achieve an error of less than one percent in less than one second. We also observe that the explicit scheme is slightly more efficient.

Table 4.5: Efficiency tests for the numerical schemes for the monodomain model in 1D.

Numerical scheme	CPU-time (seconds)	Δx	Δt	error
Operator splitting	0.91	0.00800	0.000250	0.97
Explicit	0.69	0.007692	0.000182	0.99

Efficiency tests for the numerical schemes for the monodomain model in 1D. The reference solution is computed with the explicit scheme and it has resolution $\Delta x = 1/10000$ and $\Delta t = 1/21000000$. The error is computed at $t = 0.1$ and it is measured in percent.

4.3.3 Convergence Rates in 2D

Table 4.6 and Table 4.7 shows convergence results for the numerical scheme applied to the monodomain model in 2D. The errors are computed at time

$t = 0.1$. The reference solution is computed with the explicit scheme, and it has resolution $\Delta x = 1/600$ and $\Delta t = 1/160000$. As commented above, the spatial resolution of $\Delta x = 1/600$ is relatively coarse. However, we see from Table 4.6 and Table 4.7 that the convergence rates are almost as expected. We also observe that the difference between the computed and predicted convergence rates is larger for the 2D cases. This was most likely caused by the relatively coarse spatial resolution of the reference solutions in 2D.

Table 4.6: Convergence rates for the explicit scheme for the monodomain model in 2D

Δx	Δt	error	r
0.028571	0.005000	0.0101	
0.014286	0.000250	0.0030	1.7470
0.0007143	0.000125	0.0017	0.8421

Convergence results for the explicit scheme for the monodomain model in 2D. The reference solution is computed with the explicit scheme and it has resolution $\Delta x = dy = 1/600$ and $\Delta t = 1/160000$. The error is computed at $t = 0.1$.

Table 4.7: Convergence rates for the operator splitting scheme for the monodomain model in 2D

Δx	Δt	error	r
0.016667	0.010000	0.0637	
0.083333	0.002500	0.0176	0.9288
0.004167	0.000625	0.0045	0.9754

Convergence results for the operator splitting scheme for the monodomain model in 2D. The reference solution is computed with the explicit scheme and it has resolution $\Delta x = 1/600$ and $\Delta t = 1/160000$. The error is computed at $t = 0.1$.

4.3.4 Efficiency in 2D

Table 4.8 shows the efficiency for both the explicit and semi-implicit scheme applied to the monodomain-model. We observe that the operator splitting scheme is more efficient than the explicit scheme. However, we also observe that both schemes achieve an error of less than one percent in less than one second.

Table 4.8: Efficiency tests for the operator splitting scheme for the monodomain model in 2D.

Numerical scheme	CPU-time (seconds)	Δx	Δt	error
Operator splitting	0.21	0.028571	0.002778	0.99
Explicit	0.48	0.022727	0.000556	0.98

Efficiency tests for the numerical schemes for the monodomain model in 2D. The reference solution is computed with the explicit scheme and it has resolution $\Delta x = 1/600$ and $\Delta t = 1/160000$. The error is computed at t = 0.1 and it is measured in percent.

4.4 Results for the Bidomain Model

We used the explicit scheme to compute the reference solution for the monodomain model. This choice was motivated by the fact that the explicit scheme is significantly faster than the semi-implicit scheme for a given resolution. But for the explicit scheme for the bidomain model, we use a fixed number of Jacobi iterations for each time step. Since we do not use Jacobi iterations until convergence, this error could potentially become large. This is an undesirable property for a reference solution. We therefore prefer the operator splitting scheme to compute a reference solution for the bidomain model. However, this scheme is more computationally demanding than the explicit scheme for a given resolution, especially for 2D problem with a high spatial resolution. I managed to compute a reference solution with $\Delta x = 1/500$ and $\Delta t = 1/5000$. One could argue that this resolution is too low for a reference solution. However, as we will see below, the convergence rate tests indicate that this reference solution has a satisfactory resolution.

4.4.1 Convergence Rates in 1D

We expect spatial convergence of order two and convergence in time of order one for the operator splitting scheme. Table 4.9 shows convergence tests for the operator splitting scheme for the bidomain model. The reference solution is computed with the operator splitting scheme and it has resolution $\Delta x = 1/10000$ and $\Delta t = 1/400000$. We observe that the convergence rates are close to one in time and two in space, which is as expected.

Table 4.9: Convergence rates for the operator splitting scheme for the bidomain equations in 1D

Δx	Δt	error	r
0.001961	0.010000	0.0819	
0.000980	0.002500	0.00256	0.8402
0.000490	0.000625	0.0071	0.9232
0.000245	0.000156	0.0020	0.9137

Convergence results for the operator splitting scheme for the bidomain model in 1D. The reference solution is computed with the operator splitting scheme and it has resolution $\Delta x = 1/10000$ and $\Delta t = 1/400000$. The error is computed at $t = 0.1$

It is less obvious which convergence rates we should expect for the explicit scheme for the bidomain model. As for the monodomain model, we have spatial errors from the finite element discretization and the use of a lumped mass matrix, as well as errors for the temporal discretization. In addition, we have an error from not solving the linear system to convergence, but instead applying a fixed number of Jacobi iterations. It is likely that it will depend on the time step Δt, since a smaller time step will lead to a better start value of the iterations. However, we have not analyzed this error in detail, but it is reasonable to expect a spatial convergene less than one.

Table 4.10 shows results of the convergence tests for the explicit scheme for the bidomain model when we use only one Jacobi iteration to solve the elliptic part. The reference solution has resolution $\Delta x = 1/10000$, $\Delta t = 1/400000$, and it is computed using the operator splitting scheme. The error is computed at $t = 0.1$. When we use only one Jacobi iteration to solve the elliptic part of the problem, we observe that the scheme does not converge, i.e. the error grows as the temporal and spatial resolution is increased by the same factor. Although we expected a reduced convergence rate for this scheme, the observed behavior is somewhat non-intuitive. Due to time limitations, we have not performed an analysis to identify the cause of this behaviour.

Table 4.10: Convergence rates for the explicit scheme for the bidomain equations in 1D

Δx	Δt	error	r
0.020000	0.000235	0.0248	
0.001000	0.000118	0.0296	-0.2578
0.000500	0.000059	0.0903	-1.6097
0.000250	0.000029	0.1476	-0.7091

Convergence results for the explicit scheme for the bidomain model in 1D when we use only one Jacobi iteration to solve the elliptic part. The reference solution is computed with the operator splitting scheme and it has resolution $\Delta x = 1/10000$ and $\Delta t = 1/400000$. The error is computed at t = 0.1.

4.4.2 Efficiency in 1D

Our starting point was to find a resolution for the explicit scheme which gives an error less than one percent when we solve the elliptic part of the problem with an error smaller than 1e-15 [1]. In Table 4.11, we observe that we achieve an error less than one percent with the resolution $\Delta x = 0.007142, \Delta t = 0.000143$, in 1.90 seconds.

[1] When we find the solution to the elliptic part of the problem with an error smaller than 1e-15, we use a conjugate gradient method implemented in Fenics.

Table 4.11: Efficiency tests for the numerical schemes for the bidomain equations in 1D

Method for solving elliptic part	CPU-time	Δx	Δt	error
Conjugate gradient method until convergence	1.90	0.007142	0.000143	0.97
One Jacobi iteration	1.18	0.007142	0.000143	10.22
Three Jacobi iterations	1.60	0.007142	0.000143	2.13
One Jacobi iteration	3.02	0.007142	0.000040	1.43
Operator splitting	8.63	0.004762	0.000357	0.94

Efficiency tests for the numerical schemes for the bidomain equations in 1D. The reference solution is computed with the operator splitting scheme and it has resolution $\Delta x = 1/10000$ and $\Delta t = 1/400000$. The error is computed at t = 0.1 and it is measured in percent.

Then we investigate how the explicit scheme performs when we use only one Jacobi iteration to solve the elliptic part. We computed a numerical solution using the explicit scheme with the same resolution as above ($\Delta x = 0.007142, \Delta t = 0.000143$), but this time using only one Jacobi iteration to solve the elliptic part. Table 4.11 row two shows that the error in this case is 10.22 percent and the CPU-time is 1.18 seconds. As expected, the error is larger and the CPU-time is lower when we use only one Jacobi iteration compared to when we solve the elliptic part until convergence. However, the error is about 10 times larger when we use only one Jacobi iteration, which is not a desirable property for the explicit scheme.

To explore the impact of increasing the number of Jacobi iterations on the accuracy and efficiency, we also computed the CPU-time and error for the same resolution ($\Delta x = 0.007142, \Delta t = 0.000143$), but this time using three Jacobi iterations to solve the elliptic part. If we compare row two to row three in Table 4.11, we observe that the CPU-time is increased by approximately 25 percent when the number of Jacobi iterations is increased by a factor of three. We also observe that the error has decreased by about 80 percent. Both these observations are qualitatively as we expected.

We will now explore how the accuracy and efficiency of the explicit scheme is affected by a change in the temporal resolution. We expect that an increase in the temporal resolution should decrease the error arising from using a fixed number of Jacobi iterations. Table 4.11 row four shows the results if we use one Jacobi iteration, hold the spatial resolution fixed at $\Delta x = 0.007142$, and increase the temporal resolution to $\Delta t = 0.00004$. Comparing row two to row four, we see that the error has decreased by approximately 85 percent. We also see that the CPU time has increased by more than 100 percent.

The next step is to compare the efficiency of the explicit scheme to the efficiency of the operator splitting scheme. Table 4.11 row five shows that the operator splitting scheme achieves an error of less than one percent in 8.63 seconds. Comparing row two and four to row five in Table 4.11, we observe that the operator splitting scheme performs better in terms of efficiency than the explicit scheme which uses only one Jacobi iteration. Comparing all different cases, we observe that solving the explicit part of the scheme until convergence performs best in terms of efficiency. These result indicates that using a fixed number of Jacobi iterations for solving the elliptic part of the problem is not the most efficient way of solving the bidomain equations in 1D.

4.4.3 Convergence Rates in 2D

Table 4.12 shows convergence tests for the semi-implicit scheme for the bidomain model. The error is computed at t = 0.1. The reference solution is computed with the operator splitting scheme and it has resolution $\Delta x = 1/500$, and $\Delta t = 1/5000$. A spatial resolution of $\Delta x = 1/500$ is relatively coarse. However, we observe that the convergence rates are close to one in time and two in space, which is almost as expected. Thus, this convergence test indicate that our operator splitting scheme is correctly implemented, and that the resolution $\Delta x = 1/500$, and $\Delta t = 1/5000$ is sufficient for a reference solution.

Table 4.12: Convergence rates for the operator splitting scheme for the bidomain equations in 2D

Δx	Δt	error	r
0.016667	0.012500	0.0858	
0.008333	0.003125	0.0254	0.8776
0.004167	0.000781	0.0054	1.1127

Convergence results for the operator splitting scheme for the bidomain model in 2D. The reference solution is computed with the operator splitting scheme and it has resolution $\Delta x = 1/500$ and $\Delta t = 1/5000$. The error is computed at t = 0.1

Table 4.13 shows convergence tests for the explicit scheme for the bidomain model. The error is computed at $t = 0.1$. We see that the convergence rates are negative, which was also the case in 1D.

Table 4.13: Convergence rates for the explicit scheme for the bidomain equations in 2D

Δx	Δt	error	r
0.050000	0.001538	0.0340	
0.025000	0.000769	0.0572	-0.7053
0.012500	0.000385	0.1191	-1.0568

Convergence results for the explicit scheme for the bidomain model in 2D. The reference solution is computed with the operator splitting scheme and it has resolution $\Delta x = 1/500$ and $\Delta t = 1/5000$. The error is computed at t = 0.1

4.4.4 Efficiency in 2D

Table 4.14 shows the results from the efficiency tests. The tests for the bidomain model in 2D are the same as the tests done for the 1D case. We observe that the explicit scheme is significantly more efficient than the operator splitting scheme. We also see that increasing the number of Jacobi iterations and increasing the temporal resolution for a given spatial resolution had a positive impact on the

accuracy. In contrast to the 1D case, the explicit scheme using three Jacobi iterations almost achieves an error less than one percent with a CPU-time of only 0.41 seconds. Since realistic models of the electrical activity in the heart are multidimensional, these results indicate that we can achieve speedups when solving a physiologically accurate bidomain model using an explicit scheme with a fixed number of Jacobi iterations.

Table 4.14: Efficiency tests for the numerical schemes for the bidomain equations in 2D

Method for solving elliptic part	CPU-time	Δx	Δt	error
Conjugate gradient method until convergence	1.32	0.027778	0.001429	0.93
One Jacobi iteration	0.34	0.027778	0.001429	8.34
Three Jacobi iterations	0.41	0.027778	0.001429	1.68
One Jacobi iteration	0.84	0.027778	0.000040	1.16
Operator splitting	4.53	0.025000	0.001429	0.95

Efficiency tests for the numerical schemes for the bidomain equations in 2D. The reference solution is computed with the operator splitting scheme and it has resolution $\Delta x = 1/500$ and $\Delta t = 1/5000$. The error is computed at t = 0.1 and it is measured in percent.

Chapter 5

Conclusion

In this thesis we have studied efficient numerical methods for the bidomain and monodomain equations, which describe the electrophysiology in the heart. We used a "methods of line" approach, i.e. we first applied the finite element method for a spatial discretization of the models. Then we implemented both semi-implicit and explicit schemes for temporal discretization. We used a fixed number of Jacobi iterations for the explicit scheme to solve the stationary, elliptic part of the bidomain model.

Since the operator splitting schemes use P1 elements and Godunov splitting for both the monodomain and bidomain models, we expect spatial convergence of order two and convergence in time of order one. Since the explicit scheme for the monodomain model applies mass lumping and employs the forward Euler method for temporal discretization, we expect first order convergence in time and space. The computed convergence rates mostly confirmed these theoretical predictions. However, we found that the difference between the computed and predicted convergence rates is larger for the 2D cases. This was most likely caused by the relatively coarse spatial resolution of the reference solutions in 2D.

We argued that it was reasonable to expect a spatial convergence of less than order one for the explicit scheme applied to the bidomain model, see section 4.4.1 for a discussion. However, the explicit scheme based on Jacobi iterations di-

verged when we increased the temporal and spatial resolution by the same factor, which is not a desirable property of a numerical scheme. Due to time limitations, we have not analyzed this behavior further. However, we also found that increasing the temporal resolution for given spatial resolution for the explicit scheme based on Jacobi iterations increased the accuracy.

We used the CPU-time for a numerical scheme to achieve an error less than one percent as a measure of efficiency. We found that the explicit scheme performed better than the operator splitting scheme in 1D, and that the operator splitting scheme performed better than the explicit scheme in 2D. However, since both schemes in 1D and 2D achieved an error less than one percent in less than one second, one can for practical purposes say that the schemes for the monodomain model performed comparably.

For the bidomain model in 1D, the explicit scheme which solved the elliptic part until convergence had the best efficiency. The operator splitting scheme performed better than the explicit scheme based on Jacobi iterations. For the 2D case however, the explicit scheme based on Jacobi iterations performed better than the operator splitting scheme. Since realistic models of the electrical activity in the heart are multidimensional, these results indicate that we can achieve speedups when solving a physiologically accurate bidomain model using an explicit scheme based on a fixed number of Jacobi iterations. We also found that increasing the number of Jacobi iterations and increasing the temporal resolution for a given spatial resolution had a positive impact on the accuracy of the explicit scheme based on Jacobi iterations both in 1D and in 2D. When interpreting the results regarding computational efficiency, we have to keep in mind that the operator splitting for the bidomain model is not optimally implemented, see section 3.4.3 for a discussion.

The parameter values were chosen to achieve a reasonable shape of the action potential. However, the parameter values are not physiologically accurate. The test cases were computed on simple geometries rather than physiologically accurate geometries. Hence, it is possible that more realistic parameter values

and spatial grids could have produced different empirical results.

We will now discuss extensions that could improve the efficiency and accuracy for the explicit solvers. As mentioned above, the explicit scheme based on a fixed number of Jacobi iterations diverged when we performed convergence tests. Further work could possibly investigate the source of this behavior. The forward Euler method, which was used for temporal discretization for the explicit schemes, has only first-order accuracy. A way of improving the explicit schemes is to use forward differences with a higher order of accuracy. Further, we only considered lumped mass matrices for our explicit scheme. Although mass lumping reduces the computational burden for a given resolution, it comes at the cost of reduced spatial convergence. Using a consistent mass matrix instead could influence the results. Finally, we only tried Jacobi iterations for solving the elliptic part of the bidomain model. A different iterative method or a direct method could also have produced different results.

Appendix

The scripts which can reproduce the results in this thesis are in the following repository: https://github.com/chrbjorl/master.git.

Bibliography

[1] Yves Bourgault, Marc Ethier, and Victor G LeBlanc. Simulation of electrophysiological waves with an unstructured finite element method. *ESAIM: Mathematical Modelling and Numerical Analysis*, 37(4):649–661, 2003.

[2] Marc Ethier and Yves Bourgault. Semi-implicit time-discretization schemes for the bidomain model. *SIAM Journal on Numerical Analysis*, 46(5):2443–2468, 2008.

[3] Hans Petter Langtangen. *Computational partial differential equations: numerical methods and diffpack programming*, volume 2. Springer Science & Business Media, 2013.

[4] Hans Petter Langtangen and Anders Logg. Solving pdes in minutes-the fenics tutorial volume i, 2016.

[5] Maria Murillo and Xiao-Chuan Cai. A fully implicit parallel algorithm for simulating the non-linear electrical activity of the heart. *Numerical linear algebra with applications*, 11(2-3):261–277, 2004.

[6] Steffan Puwal and Bradley J Roth. Forward euler stability of the bidomain model of cardiac tissue. *IEEE transactions on biomedical engineering*, 54(5):951–953, 2007.

[7] Jack M Rogers and Andrew D McCulloch. A collocation-galerkin finite element model of cardiac action potential propagation. *IEEE Transactions on Biomedical Engineering*, 41(8):743–757, 1994.

[8] Joakim Sundnes, Glenn Terje Lines, Xing Cai, Bjørn Frederik Nielsen, Kent-Andre Mardal, and Aslak Tveito. *Computing the electrical activity in the heart*, volume 1. Springer Science & Business Media, 2007.

[9] Joakim Sundnes, Glenn Terje Lines, and Aslak Tveito. An operator splitting method for solving the bidomain equations coupled to a volume conductor model for the torso. *Mathematical biosciences*, 194(2):233–248, 2005.

[10] Joakim Sundnes, Bjørn Fredrik Nielsen, Kent Andre Mardal, Xing Cai, Glenn Terje Lines, and Aslak Tveito. On the computational complexity of the bidomain and the monodomain models of electrophysiology. *Annals of biomedical engineering*, 34(7):1088–1097, 2006.

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

[12] Edward J Vigmond, Felipe Aguel, and Natalia A Trayanova. Computationally efficient methods for solving the bidomain equations in 3d. In *Engineering in Medicine and Biology Society, 2001. Proceedings of the 23rd Annual International Conference of the IEEE*, volume 1, pages 348–351. IEEE, 2001.