MULTIGRID METHODS FOR THE BIDOMAIN EQUATIONS

Yanni Lai

A dissertation submitted to the faculty at the University of North Carolina at Chapel Hill in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Department of Mathematics in the College of Arts and Sciences.

Chapel Hill
2019

ABSTRACT

Yanni Lai: Multigrid Methods for the Bidomain Equations
(Under the direction of Boyce Griffith)

The study of cardiac electrophysiology has many applications in medical practice. One important model is the bidomain equations. In the thesis, the bidomain equations for the muscle and for the muscle and the bath are considered. By implementing multigrid algorithms as the preconditioner, we explore the block factorization approach for solving the bidomain equations.

The dissertation consists two parts, aiming to present the biological background and discretization for the bidomain equations, as well as the multigrid algorithms. In the first part, we present the derivation of the formula of bidomain equations, the finite difference and finite element discretization for the bidomain system, and semi-implicit time stepping.

In the second part, we study the key facts of both geometric multigrid and algebraic multigrid method. We consider the with and without fibrosis cases. We implement the two multigrid methods as both the solver for the bidomain system and the preconditioner for the block factorization approach, and conclude that block factorization works efficiently, especially compared with the performance of the algebraic multigrid solver. We also test the block factorization with algebraic multigrid preconditioner on a realistic three-dimensional geometry, and obtain only a small increase in solver iterations as the mesh becomes finer.

We discuss useful extensions of this block factorization approach on solving the bidomain system. Since algebraic multigrid works best for Poisson-like problems, we can factorize the original matrix into blocks with poisson like form, and apply algebraic multigrid as preconditioner to each block to achieve good convergence.

ACKNOWLEDGEMENTS

First and foremost, I would like to thank my thesis advisors, Dr. Boyce Griffith, whose guidance has been indispensable for my academic studies I am especially grateful for the meaningful suggestions he made to my research, as well as his extraordinary perspective on computational mathematics, which will no doubt shape my own philosophy as I move forward.

I would like to thank my thesis committee members: Dr. Jingfang Huang, Dr. David Adalsteinsson, Dr. Katie Newhall, and Dr. Simone Rossi. Special thanks to them for the support and suggestions. Without their help, my achievements would not be possible.

I would also like to thank the tremendous support and encouragement I received from the math department at the University of North Carolina, especially from Dr. Richard McLaughlin, Dr. Laura Miller, and Ms. Laurie Straube.

I am indebted as well to my fellow students, with whom I have shared this incredible journey. Special thanks go to Dangxing Chen, Yunyan He, Fuhui Fang, Jason Pearson, Shengjie Chai, and Tianxiao Sun. I would also like to express my deep appreciation to all the wonderful faculty members and students at the University of North Carolina.

Last but not least, my parents and my grandmother hold all my gratitude for their patience and for the love they give to me. This thesis is dedicated to them, together with Jianxiao Liu, Yiran Hua, and Yifu Yang.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

AMG	algebraic multigrid method. 15
AV	atrioventricular. 3
CG	conjugate gradient method. 18
CNAB	Crank-Nicolson-Adams-Bashfort. 17
FDM	finite-difference method. 13
FEM	finite-element method. 13
FGMRES	flexible GMRES. 19
FVM	finite-volume method. 13
GMG	geometric multigrid method. 15
ILU	incomplete LU. 18
ODE	ordinary differential equations. 4
PCG	preconditioned conjugate gradient. 18
SA	sinoatrial. 3
SOR	Successive Over-Relaation. 19

CHAPTER 1

Introduction

1.1 Biological Background

1.1.1 Cardiac Electrophysiology

The study of cardiac electrophysiology is concerned with the electrical activity of the heart muscle. In a cell, the cell membrane separates its interior, which is the intracellular space, and its exterior, which is the extracellular space. Through the membrane there are proteins called ion channels. The difference in electric charge between the inside and outside of the cell across a cellular membrane is the transmembrane voltage difference. When the transmembrane voltage difference changes, the ion channels of an excitable cell can become activated, and ions can flow across the cell membrane through the channels [2]. The nearly constant transmembrane voltage difference under resting conditions is called the resting potential [3].

The action potential is the rapid rise and eventual fall of the transmembrane voltage. The action potential consists of five phases: upstoke, partial re-polarization, plateau, re-polarization, resting. A cell under resting conditions has the capacity for a change of transmembrane voltage to more positive, which is called depolarization. The action potential begins with the upstroke, which is the rapid depolarization. During the upstroke, there is a rapid activation of sodium channels. Sodium ions flow into the cell through these channels, adding positive charge to the inside of the cell. When the upstroke ends, sodium channels close rapidly. After the upstroke, the potassium channels open, allowing the potassium ions flowing outward of the cell, making membrane potential more negative, which is the partial re-polarization phase. The plateau phase, which is characterized by a balance between the inward calcium current and the outward potassium current, happens after the partial re-polarization. Following the plateau, the cell experiences the re-polarization phase, in which the calcium channels close while the potassium channels remain open. In this phase, the potassium ions keep moving out of the cell, causing the transmembrane voltage to more negative.

After re-polarization, the cell return to its resting potential [4].

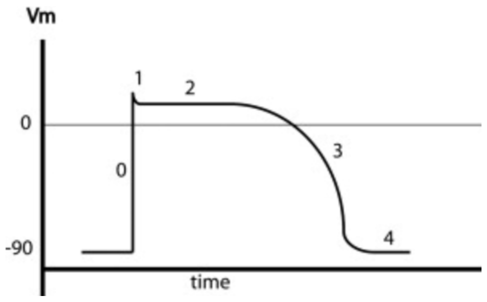

Figure 1.1: Five phases of the action potential: phase 0 is upstroke, phase 1 is partial re-polarization, phase 2 is plateau, phase 3 is re-polarization, and phase 4 is resting. *https://www.sciencedirect.com/topics/immunology-and-microbiology/heart-contraction*

1.1.2 The Structure of Cardiac Muscle

In the heart, the upper chambers are called the atria, and the lower chambers are called the ventricles. The artria act as primer pumps that ensure blood flow to the ventricles, whereas the ventricles are powerful pumping chambers. The left ventricle pumps oxygenated blood from the lungs throughout the body, and the right ventricle pumps deoxygenated blood back to the lungs.

A myocyte is the spindle-shaped cell found in muscle tissue. Cardiacmyocyte is the myocyte that account for most of the mass of the atrial and ventricular muscle. A typical cardiomyocyte is approximately 100 μm in length and 10-25 μm in diameter. Cardiomyocytes have the ability to contract, which allows the heart to pump. Each cardiomyocyte's contraction is in coordination with its neighbors. In between individual cardiomyocytes, there are intercalated disks to join them together. The gap junctions, which are small ionic channels in cylinder shape, are contained in the intercalated disks [5]. Most of the gap junctions in cardiac tissue are coupled end-to-end. Gap junctions allow the transmission of ionic currents and the spread of action potentials from cell to cell. The length of the gap junctions is about 2-12 nm, with a diameter of 2 nm [6].

There is spatial variation in the arrangement and morphology of cardiacmyocytes in the heart. Specifically, those in the atria are different from those in the ventricle [7]. Atrial myocytes have different gene expression patterns regarding the transcription factors, and different fibrous proteins from the ventricular myocytes. The distribution of ion channels also differs in different locations, leading to the regional differences of action potential shape and conduction velocity, which is the velocity that cardiac muscle cells send signals to heart muscle to cause it to contract [8].

The network of macromolecules in cardiac tissue is called the extracellular matrix [9]. The main structural protein in the extracellular matrix is the collagen. Cardiscyocytes are surrounded by the extracellular matrix. The extracellular matrix provides structured biochemical support around the cells, and is important for cells' reorganization and differentiation. Collagen densities vary with tissue type. For example, in the ventricular myocardium, which is the thick middle layer of the heart wall, the density of collagen is higher compared with that in the inner and outer layers [7].

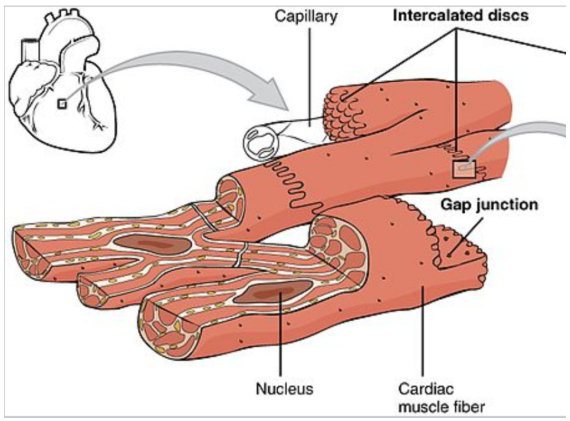

Figure 1.2: Cardiacmyocytes have a cylindrical shape. They are connected with each other by the intercalated disks. There are gap junctions formed of small ionic channels contained in the intercalated disks. *https://courses.lumenlearning.com/austincc-ap1/chapter/cardiac-muscle-tissue*

1.1.3 Cardiac Conduction System

In the cardiac conduction system, a group of cells send signals to the heart muscle, leading to the contraction [2]. There are five major components of the cardiac conduction system, including the sinoatrial (SA) node, the muscular heart tissue, the atrioventricular (AV) node, the AV bundle, and the Purkinje fibers. Located in the upper part of the wall of the right atrium, the SA node generates the action potential to stimulate the contraction. The excitation spreads through the atrial myocardium and reaches the AV node located in the lower part of the right atrium. Upon receiving the signal, the AV node fires, and excitation spreads down the bundle brunches in the ventricles. Finally, the Purkinje fibers distribute the extraction to the ventricular myocardium [2].

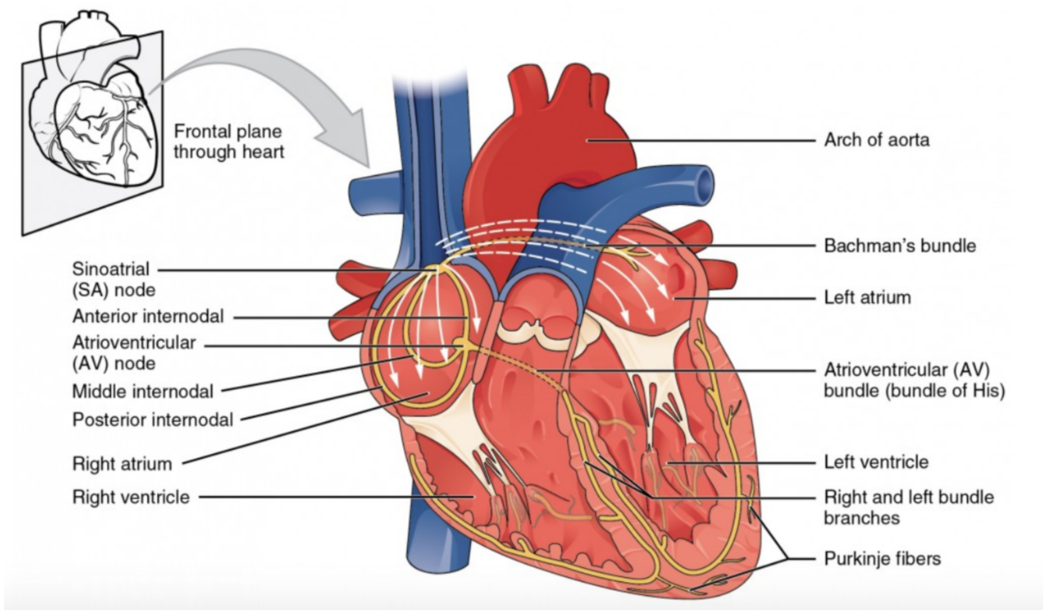

Figure 1.3: There are five major components of the cardiac conduction system, including the SA node, the muscular heart tissue, the AV node, the AV bundle, and the Purkinje fibers. *https://courses.lumenlearning.com/suny-ap2/chapter/cardiac-muscle-and-electrical-activity*

The nodal cells refer to the cells within the SA and AV nodes. While having an action potential, these cells have no true resting potential. There are three phases for the SA nodal action potentials: the spontaneous depolarization that triggers the action potential, the depolarization of the action potential, and the re-polarization. The cycle is spontaneously repeated. The membrane potential changes in different phases corresponds primarily to the movements of calcium and potassium ions through the ion channels [10].

1.2 Discrete Cellular Models

There are two basic types of models of cardiac electrophysiology: discrete models and continuous models. In discrete models, cardiac tissue is characterized based on individual cells, while in continuous models, cardiac tissue is treated as a functional syncytium (cells are linked with each other and are viewed as a whole system). Discrete models of cardiac tissue include simple cellular automaton models, coupled map lattices [11], and lattices of the system of coupled ordinary differential equations (ODE) [12].

In cellular automaton models, each cell is coupled to its neighbors, and has a finite number of states. The state of cell in each time step is updated based on its state in the previous time step, as well as the state of its neighbors. The same transition rule applies simultaneously to every cell. This kind of model is easy to implement, and is computationally inexpensive [13]. One revision of the cellular automaton models is the coupled map lattices, which involve the continuous states. The states in these models are decided by the interactions within a lattice. To allow modeling anisotropic propagation, the coupling strength of each interaction is different [14]. A further development of the previous models is to add the ODEs. With this kind of model, the detailed tissue architecture at the cell level can be modeled through the ODEs. [15]. However, this approach is computationally expensive. Also, extra information is needed to complete these models: the composition of the extracellular space, the information of cell size and capacitance, and the conductance of the gap junctions [16].

1.3 Continuous Models

In the continuous models, we view cardiac tissue as a single unit composed of electrically connected cells. The classic bidomain and monodomain models are two examples of continuous models of cardiac muscle. The bidomain system represents cardiac tissue as a functional unit comprised of intracellular and extracellular compartments. We assume the two compartments to be continuous and overlapping, and are separated by a continuous cell membrane [17].

1.3.1 Derivation

The bidomain model is based on a generalized version of Ohm's law, which states that in a conducting body, the current density **J** at a specific location is proportional to the electric field **E** at that location,

$$\mathbf{J} = \sigma \mathbf{E}, \tag{1.3.1}$$

where σ is the conductivity of the material [18]. The electric field **E** is defined as the gradient of a scalar potential ϕ

$$\mathbf{E} = -\nabla \phi. \tag{1.3.2}$$

Thus

$$\mathbf{J} = -\sigma \nabla \phi. \tag{1.3.3}$$

The intracellular and extracellular current density $\mathbf{J}_\mathrm{i}$ and $\mathbf{J}_\mathrm{e}$ are defined by:

$$\begin{aligned} \mathbf{J}_\mathrm{i} &= -\sigma_\mathrm{i} \nabla \phi_\mathrm{i}, \\ \mathbf{J}_\mathrm{e} &= -\sigma_\mathrm{e} \nabla \phi_\mathrm{e}, \end{aligned} \tag{1.3.4}$$

where σ_i and σ_e represent the intracellular and extracellular conductivity tensors respectively, and ϕ_i and ϕ_e are the electrical potential in the intracellular and extracellular space. The conductivity tensors σ_i and σ_e account for the anisotropy of cardiac tissue.

Assume there is no external source of charge, for a volume Ω, and denote the surface of Ω as Γ. Then

$$\int_\Gamma \mathbf{n} \cdot (\mathbf{J}_\mathrm{i} + \mathbf{J}_\mathrm{e}) dS = 0. \tag{1.3.5}$$

By the divergence theorem, we have

$$\int_\Omega \nabla \cdot (\mathbf{J}_\mathrm{i} + \mathbf{J}_\mathrm{e}) d\vec{x} = 0. \tag{1.3.6}$$

As the above equation should hold for all specific volumes Ω in the domain, thus we have

$$\nabla \cdot (\mathbf{J}_\mathrm{i} + \mathbf{J}_\mathrm{e}) = 0. \tag{1.3.7}$$

As a consequence of Kirchhoff's law, any change in intracellular or extracellular current should be due to the transmembrane current (I_t)

$$\nabla \cdot \mathbf{J}_\mathrm{i} = I_\mathrm{t} = -\nabla \cdot \mathbf{J}_\mathrm{e}. \tag{1.3.8}$$

Substitute $\mathbf{J}_\mathrm{i}$ and $\mathbf{J}_\mathrm{e}$ in equations (1.3.4) to equation (1.3.8):

$$\begin{aligned} \nabla \cdot (\sigma_\mathrm{i} \nabla \phi_\mathrm{i}) &= I_\mathrm{t}, \\ \nabla \cdot (\sigma_\mathrm{e} \nabla \phi_\mathrm{e}) &= -I_\mathrm{t}. \end{aligned} \tag{1.3.9}$$

The transmembrane current (I_t) equals to the sum of a capacitive current (I_c), a resistive current (I_ion), and a stimuli applied across the membrane (I_sti, take the positive to be the outward direction) [19]

$$I_\mathrm{t} = \chi(I_\mathrm{c} + I_\mathrm{ion}) - I_\mathrm{sti}, \tag{1.3.10}$$

where χ is the surface area-to-volume ratio of the cell, and I_ion represents the current flowing through the ion channels [17]. The capacitance current is modeled by

$$I_\mathrm{c} = C_\mathrm{m} \frac{\partial V}{\partial t}. \tag{1.3.11}$$

Substituting I_c in (1.3.11) into (1.3.10), we have

$$I_\mathrm{t} = \chi(C_m \frac{\partial V}{\partial t} + I_\mathrm{ion}) - I_\mathrm{sti}. \tag{1.3.12}$$

Equating the first equation of (1.3.9) and (1.3.12) yields the first parabolic equation of the bidomain system

$$\chi(C_m \frac{\partial V}{\partial t} + I_\mathrm{ion}) - I_\mathrm{sti} = \nabla \cdot (\sigma_\mathrm{i} \nabla \phi_\mathrm{i}). \tag{1.3.13}$$

Equating the second equation of (1.3.9) and (1.3.12) yields the second parabolic equation of the bidomain system

$$-\chi(C_m \frac{\partial V}{\partial t} + I_\mathrm{ion}) + I_\mathrm{sti} = \nabla \cdot (\sigma_\mathrm{e} \nabla \phi_\mathrm{e}). \tag{1.3.14}$$

1.3.2 Two Forms

Combine equation (1.3.13) and (1.3.14). For the applied stimuli, take the positive to be the direction into the intracellular space, and I_{sti} in equation (1.3.13) should be positive ($I_{\mathrm{i}}^{(\mathrm{vol})}$), while I_{sti} in equation (1.3.14) should be negative ($-I_{\mathrm{e}}^{(\mathrm{vol})}$). The parabolic-parabolic form of the bidomain equations is:

$$\nabla \cdot (\sigma_{\mathrm{i}} \nabla \phi_{\mathrm{i}}) = \chi(C_{\mathrm{m}} \frac{\partial V}{\partial t} + I_{\mathrm{ion}}(\mathbf{u}, V)) - I_{\mathrm{i}}^{(\mathrm{vol})}, \tag{1.3.15}$$

$$\nabla \cdot (\sigma_e \nabla \phi_e) = -\chi(C_{\mathrm{m}} \frac{\partial V}{\partial t} + I_{\mathrm{ion}}(\mathbf{u}, V)) - I_{\mathrm{e}}^{(\mathrm{vol})}. \tag{1.3.16}$$

In equation (1.3.15), $\mathbf{u}$ is a set of cell-level variables, such as ionic concentrations. Functional forms of I_{ion} is obtained from specific electrophysiological cell models [20]. Equation (1.3.15) represents the local conservation of current in the intracellular region, and equation (1.3.16) represents that in the extracellular region.

The first equation of the parabolic-elliptic form of bidomain system is equation (1.3.16), with the substitution of $\phi_{\mathrm{i}} = V + \phi_{\mathrm{e}}$. We obtain the second equation by adding equation (1.3.15) and (1.3.16) together, such that

$$\begin{cases} \nabla \cdot (\sigma_{\mathrm{i}} \nabla \phi_{\mathrm{i}}) + \nabla \cdot (\sigma_e \nabla \phi_e) = \nabla \cdot (\sigma_{\mathrm{i}} \nabla (V + \phi_{\mathrm{e}})) + \nabla \cdot (\sigma_e \nabla \phi_e), \\ \chi(C_{\mathrm{m}} \frac{\partial V}{\partial t} + I_{\mathrm{ion}}(\mathbf{u}, V)) - \chi(C_{\mathrm{m}} \frac{\partial V}{\partial t} + I_{\mathrm{ion}}(\mathbf{u}, V)) = 0, \\ -I_{\mathrm{i}}^{(\mathrm{vol})} - I_{\mathrm{e}}^{(\mathrm{vol})} = -(I_{\mathrm{i}}^{(\mathrm{vol})} + I_{\mathrm{e}}^{(\mathrm{vol})}) = -I_{\mathrm{total}}^{(\mathrm{vol})}, \end{cases} \tag{1.3.17}$$

where $I_{\mathrm{total}}^{(\mathrm{vol})}$ is the sum of applied stimuli. The parabolic-elliptic form is the one implemented in the numerical tests considered herein. The complete form of the parabolic-elliptic bidomain equations is

$$\nabla \cdot (\sigma_i \nabla (V + \phi_{\mathrm{e}})) = \chi(C_{\mathrm{m}} \frac{\partial V}{\partial t} + I_{\mathrm{ion}}(\mathbf{u}, V)) - I_{\mathrm{i}}^{(\mathrm{vol})}, \tag{1.3.18}$$

$$\nabla \cdot ((\sigma_{\mathrm{i}} + \sigma_{\mathrm{e}}) \nabla \phi_{\mathrm{e}}) = -\nabla \cdot (\sigma_{\mathrm{i}} \nabla V) - I_{\mathrm{total}}^{(\mathrm{vol})}. \tag{1.3.19}$$

We will assume $I_{\mathrm{total}}^{(\mathrm{vol})} = 0$, which corresponds to having $I_{\mathrm{e}}^{(\mathrm{vol})} = -I_{\mathrm{i}}^{(\mathrm{vol})}$, meaning to apply an extracellular stimulus equal and opposite to the intracellular stimulus. The positive $I_{\mathrm{i}}^{(\mathrm{vol})}$ and negative $I_{\mathrm{e}}^{(\mathrm{vol})}$ corresponds to the conservation of current by injecting current into the intracellular space while simultaneously pulling the same amount out of the extracellular space [20].

The properties of equation (1.3.18) is similar to a parabolic PDE, specifically, a reaction-diffusion system. Equation (1.3.19) resembles a boundary value problem, which is an elliptic PDE.

1.3.3 Boundary Condition

To solve the bidomain model, we should add the boundary conditions. Suppose the model is defined on a volume Ω with Γ that at the boundary of the tissue. Similar as in section (1.3.1), let $\mathbf{J}_\mathrm{i}$ and $\mathbf{J}_\mathrm{e}$ be the intracellular and extracellular current density across the boundary respectively. Denoting $I_\mathrm{i}^{(\mathrm{surf})}$ and $I_\mathrm{e}^{(\mathrm{surf})}$ as the intracellular and extracellular currents per unit area applied across the boundary, we have

$$\begin{aligned} \mathbf{n} \cdot \mathbf{J}_\mathrm{i} &= I_\mathrm{i}^{(\mathrm{surf})}, \\ \mathbf{n} \cdot \mathbf{J}_\mathrm{e} &= I_\mathrm{e}^{(\mathrm{surf})}, \end{aligned} \tag{1.3.20}$$

where $\mathbf{n}$ denotes the outward normal to the boundary. Substituting $\mathbf{J}_\mathrm{i}$ and $\mathbf{J}_\mathrm{e}$ in (1.3.20) with those in (1.3.4), the boundary conditions are

$$\mathbf{n} \cdot (\sigma_\mathrm{i} \nabla (V + \phi_\mathrm{e})) = I_\mathrm{i}^{(\mathrm{surf})}, \tag{1.3.21}$$

$$\mathbf{n} \cdot (\sigma_\mathrm{e} \nabla \phi_\mathrm{e}) = I_\mathrm{e}^{(\mathrm{surf})}. \tag{1.3.22}$$

1.3.4 Bidomain System with Bath

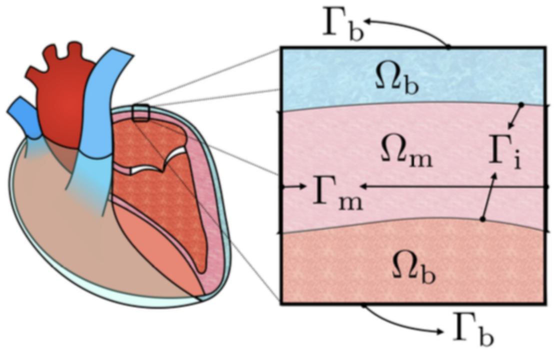

Figure 1.4: Domain with a myocardium subregion (Ω_m) and two bath regions (Ω_b). The boundaries of only the myocardium are denoted as Γ_m, those of only the bath are denoted as Γ_b, and those between the myocardium and the bath are denoted as Γ_i.*https://www.frontiersin.org/articles/10.3389/fphys.2018.01344/full*

Consider adding a conductive bath besides the myocardium, and denote the bath domain as Ω_b (figure (1.4)). In this case, the transmembrane voltage V is defined on the muscle part only, the extracellular potential ϕ_e in the muscle is defined on the muscle and the boundary between muscle and bath, and the extracellular potential ϕ_b in the bath is defined on the bath part and the boundary between muscle and bath. In Ω_b, the current density $\mathbf{J}_b$ is

$$\mathbf{J}_b = -\sigma_b \nabla \phi_b, \tag{1.3.23}$$

where σ_b is the conductivity in the bath region.

Assume there is no external source of charge, which means that for a volume Ω_b, the total current entering it equals that leaves it. Denoting the surface of the volume as Γ_b and the outward surface normal as $\mathbf{n}$, we have that

$$\int_{\Gamma_b} \mathbf{n} \cdot \mathbf{J}_b dS = 0. \tag{1.3.24}$$

Using the divergence theorem, we have

$$\int_{\Omega_\mathrm{b}} \nabla \cdot \mathbf{J}_\mathrm{b} d\vec{x} = 0. \tag{1.3.25}$$

As the above equation should hold for all specific volumes Ω_b in the domain, thus we have

$$\nabla \cdot \mathbf{J}_\mathrm{b} = 0. \tag{1.3.26}$$

Substituting $\mathbf{J}_\mathrm{b}$ in equations (1.3.23)—(1.3.26), ϕ_b satisfies:

$$\nabla \cdot (\sigma_\mathrm{b} \nabla \phi_\mathrm{b}) = 0. \tag{1.3.27}$$

Let Γ_i denote the boundary between muscle and bath. The boundary conditions for the muscle part are

$$\begin{aligned} \mathbf{n} \cdot \mathbf{J}_\mathrm{i} &= I_\mathrm{i}^{(\mathrm{surf})}, && \text{on } \Gamma_\mathrm{m}, \\ \mathbf{n} \cdot \mathbf{J}_\mathrm{e} &= I_\mathrm{e}^{(\mathrm{surf})}, && \text{on } \Gamma_\mathrm{m} \setminus \Gamma_\mathrm{i}. \end{aligned} \tag{1.3.28}$$

The boundary condition for the bath part is

$$\mathbf{n} \cdot \mathbf{J}_\mathrm{b} = I_\mathrm{b}^{(\mathrm{surf})}, \qquad \text{on } \Gamma_\mathrm{b} \setminus \Gamma_\mathrm{i}. \tag{1.3.29}$$

The boundary conditions on the boundary between the muscle and the bath are

$$\begin{aligned} \mathbf{n} \cdot \mathbf{J}_\mathrm{e} &= -\mathbf{n} \cdot \mathbf{J}_\mathrm{b}, && \text{on } \Gamma_\mathrm{i} \text{ and} \\ \phi_\mathrm{e} &= \phi_\mathrm{b}, && \text{on } \Gamma_\mathrm{i}. \end{aligned} \tag{1.3.30}$$

Thus the boundary conditions for the bidomain with bath problem are

$$\mathbf{n}\cdot(\sigma_\mathrm{i}\nabla(V+\phi_\mathrm{e})) = I_\mathrm{i}^{(\mathrm{surf})}, \qquad \text{on } \Gamma_\mathrm{m}, \tag{1.3.31}$$

$$\mathbf{n}\cdot(\sigma_\mathrm{b}\nabla\phi_\mathrm{e}) = I_\mathrm{e}^{(\mathrm{surf})}, \qquad \text{on } \Gamma_\mathrm{m}\setminus\Gamma_\mathrm{i}, \tag{1.3.32}$$

$$\mathbf{n}\cdot(\sigma_\mathrm{b}\nabla\phi_\mathrm{b}) = I_\mathrm{b}^{(\mathrm{surf})}, \qquad \text{on } \Gamma_\mathrm{b}\setminus\Gamma_\mathrm{i}, \tag{1.3.33}$$

$$\mathbf{n}\cdot(\sigma_\mathrm{e}\nabla\phi_\mathrm{e}) = -\mathbf{n}\cdot(\sigma_\mathrm{b}\nabla\phi_\mathrm{b}), \quad \text{on } \Gamma_\mathrm{i}, \tag{1.3.34}$$

$$\phi_\mathrm{e} = \phi_\mathrm{b}, \qquad \text{on } \Gamma_\mathrm{i}. \tag{1.3.35}$$

1.3.5 The Monodomain Model

Anisotropy means that the electrical properties of cardiac tissue are different in different directions. Assuming the the intracellular and extracellular regions to be equally anisotropic, we can modify the bidomain system to the monodomain system. For example, we assume the conductivity in the extracellular space to be proportional to that of the intracellular space:

$$\sigma_\mathrm{e} = \lambda\sigma_\mathrm{i}, \tag{1.3.36}$$

with ratio λ. If setting $I_\mathrm{total}^{(\mathrm{vol})}$ to be zero as mentioned in section 1.3.2, the equation (1.3.19) can be written as

$$\nabla\cdot(\sigma_\mathrm{i}\nabla\phi_\mathrm{e}) = -\frac{1}{1+\lambda}\nabla\cdot(\sigma_\mathrm{i}\nabla V). \tag{1.3.37}$$

Substituting equation (1.3.37) into equation (1.3.18) with $I_i^{(\mathrm{vol})} = 0$

$$\nabla\cdot\frac{\lambda}{1+\lambda}(\sigma_\mathrm{i}\nabla V) = \chi(C_\mathrm{m}\frac{\partial V}{\partial t} + I_\mathrm{ion}(\mathbf{u},V)). \tag{1.3.38}$$

Letting $\sigma = \frac{\lambda}{1+\lambda}\sigma_\mathrm{i}$, the final form of monodomain equation is

$$\nabla\cdot(\sigma\nabla V) = \chi(C_\mathrm{m}\frac{\partial V}{\partial t} + I_\mathrm{ion}(\mathbf{u},V)). \tag{1.3.39}$$

The boundary condition of this system is

$$\mathbf{n} \cdot (\sigma \nabla V) = 0, \tag{1.3.40}$$

assuming zero flux across the boundary.

From a computational standpoint, the major difference between the monodomain and bidomain equations is that the monodomain equations does not include the elliptic constraint (1.3.19). The presence of the elliptic constraint complicates the solution of the bidomain equations. From a modeling standpoint, accounting for extracellular currents, as done in the bidomain equations but not in the monodomain equations, is necessary to describe extracellular current sources, as in defibrillation, and to describe the details of current flow through electrodes.

1.4 Spatial Discretization

To discretize the Bidomain system, the finite-difference method (FDM), the finite-element method (FEM) and finite-volume method (FVM) are most commonly applied [21]. The FEM and FVM solve the weak form of the governing equations while the FDM solve the strong form. The advantage of using the weak form is that the boundary conditions can be easily imposed.

The FDM approximates the spatial derivatives of the continuous equations by difference quotients. With this method, the domain is usually divided into uniform grids. The approximation of the partial derivatives at each point are obtained from a truncated Taylor's series expansion of the dependent variable in terms of the values of its neighbors. This method is commonly applied to structured meshes. FDM has been widely implemented for discretizing the monodomain and bidomain equations [22]. Previous work [23] compared monodomain and bidomain models for action potential propagation using FDM. They concluded that in the absence of applied currents, the differences of the activation propogation between monodomain and bidomain models are extremely small if the monodomain model is discretized with high-resolution grids. Saleheen [24] presented the FDM for bidomain system with an inhomogeneous anisotropic tissue on a cubic uniform mesh. Specifically, the entries of the conductivity tensor matrix of tissue are functions of the direction of the fiber rotation. Another study [25] presented a higher order finite difference scheme for solving the monodomain equation. In this study, the authors expanded the transmembrane potential in terms of Lagrangian interpolating polynomials. By differentiating the polynomial expansion, they obtained the finite difference approximation, and the order of the approximation can vary according to the order of the polynomial expansion. They tested the higher order discretization scheme for the monodomain system on an idealized cubic geometry. For more complex geometries, in Sharma et al. [26], the authors solved the bidomain equations on a complex fiber geometry with the FDM. Huiskamp [27] discretized the model of the ventricle representing the myocardium of a dog using FDM. However, this creates jagged edges on curved boundaries, which influences the calculation of the boundary current flows. The advantage of FDM is the simplicity for implementation. But for irregular geometries and non-uniform meshes, FDM is difficult to apply.

The FEM is also a popular method for the monodomain and bidomain equations. With this method, the cardiac region is divided into elements. The elements can have non-uniform size and shape. The solution is approximated by interpolating nodal values for each element using basis functions. A previous study [28] applied FEM for solving the monodomain model to simulate the propagation of the excitation in the cardiac tissues. The ionic currents in this work were expressed by a modified FitzHugh-Nagumo model. Stienbach and Yang [29] studied a space-time FEM for the bidomain equations. In their work, The discretization was based on a space-time variational formulation involving both piecewise and continuous finite elements in the spatial and temporal directions. Seemann et al. [30] proposed a framework using the scientific computing library PETSc to pre-condition and solve the bidomain equation with FEM discretization. In addition, in Dal et al. [31], a fully implicit FEM algorithm for the bidomain equations were presented. Compared with FDM, FEM can be applied to more complex geometries [32]. In Costa et al. [33], FEM discretization was implemented for modeling the fibrotic clefts in the heart, which are microstructural discontinuities that disrupt the intracellular matrix. The authors developed a discontinuous finite element approach for discretizing the bidomain equations. They compared the new approach with the traditional high resolution continuous finite element models, and claimed the new method to be significantly more computationally efficient. Compared with FDM, FEM is more difficult to implement in terms of programming. To resolve these problems, a new method named finite element derived finite difference method was developed for solving the bidomain equations [34]. In this method, a FDM mesh is created over the FE geometry. Another limitation of FEM is indicated by Trew et al. [35], which mentioned that FEM imposes substantial computational costs especially if the bidomain system is solved in a discontinuous domain. To solve this problem, FVM can be considered.

The FVM considers the small volume surrounding each node point on a mesh. With this method, the volume integrals in a PDE that contains a divergence term are converted to surface integrals by applying the divergence theorem. These divergence terms are considered as fluxes at the surface of each finite volume. FVM can be easily formulated for unstructured meshes, and is popular for the bidomain and monodomain system with complex meshes. Coudiere et al. [36] analyzed the stability and convergence of FVM for the bidomain system with two time-stepping methods. A previous work [37] presented solving the bidomain equations on complex geometries with fibre rotation with FVM. In Penland et al. [38], unstructured FVM was implemented for the bidomain equations. Trew et al. [35] developed a FVM for bidomain electrical activation in discontinuous cardiac tissue. They considered modeling the cleavage planes, in which case the FVM method is desirable, since no-flux boundary conditions can be easily imposed. They mentioned that FVM has advantages over FEM for modeling cleavage planes, since FEM formulation represents cleavage planes as whole element units and thus a cleavage plane cannot have a thickness less than the mesh resolution. However, the FVM is a conservative discretization, since the flux entering a given volume is assumed to be identical to that leaving the volume.

In the numerical simulations in this thesis, I will use FDM for the geometric multigrid method (GMG) with simple two-dimensional (2D) geometries. I will use FEM for the algebraic multigrid method (AMG) with 2D and three-dimensional (3D) geometries.

1.5 Time Discretization

The choice of time-stepping methods for the bidomain system has a strong impact on the computational time and stability. In numerical implementation, if a direct computation of the dependent variables can be made based on known values, the method is called explicit. For example, the forward-Euler method is one of the basic explicit methods. With forward-Euler, the new solution is updated based on the derivative and the solution of the previous time-step. In contrast, when the solution is obtained by solving equations involving the current state, it is an implicit method. A basic example of this kind of methods is the backward-Euler method. With backward-Euler, the new solution is updated based on the derivative of the current step and the solution of the previous step.

Low order explicit time-stepping methods are very popular for solving the bidomain equations. In Puwal et al. [39], the authors calculated the stability of the 2D homogeneous anisotropic bidomain model discretized using FDM with forward-Euler. In Muzikant et al. [40], a modeling study of the bidomain system considering the fiber orientation on a 3D mesh was presented. In this study, a two-step explicit method was used. The first step is to compute a new extracellular potential with the elliptic equation and the transmembrane potential at the previous time step. The second step is to apply the explicit forward-Euler method to update the transmembrane potential to the current time-step with the parabolic equation. Santos et al. [41] presented an explicit three-step scheme for solving the transmembrane potential with the parabolic PDE, and used the forward-Euler to solve for the extracellular potential. Explicit methods are easy to implement. However, they suffer a limitation in the size of time-step due to the stability issue [42].

Some research used fully implicit methods to solve the bidomain equations. In Ethier et al. [43], the propogation of electrical potential waves with the bidomain model with the ODEs was analyzed. Different implicit time-stepping methods of order 1 (backward-Euler) and 2 (implicit Gear) were presented to discretize the bidomain system. According to the research, even with very fine grids, backward-Euler can hardly provide a wave speed error below 1%. It is only by reducing the time-step as small as that used for the forward-Euler, that the backward-Euler method provides accurate results. In Murillo et al. [44], the implicit backward-Euler method was implemented to solve the bidomain equations on a 2D square discretized uniformly with the FDM. Another study [45] also used the implicit second-order Gear method for the anisotropic bidomain model discretized on an unstructured grids. In addition, Munteanu and Pavarino [46] presented a parallel bidomain solver with the implicit backward-Euler method. Implicit methods have a much weaker limitation on the time-step size in terms of stability. But for bidomain equations, because they may involve a large system of nonlinear equations, the I_{ion} term, solving the equations involves simultaneously updating the solutions for that large system of nonlinear equations at every time-step, which is computationally expensive.

To take the advantage of the stability of implicit methods and the simplicity of the explicit methods for nonlinear problems, we consider semi-implicit methods. For solving the bidomain equations, these methods treat the I_{ion} term explicitly and the other terms implicitly. These methods are more stable than the explicit methods, and are less computationally expensive compared with the fully implicit methods, since they require the solution of a linear system of equations at each time-step, instead of a non-linear system. Multiple semi-implicit methods for bidomain equations were introduced in Ethier et al. [43]. In that work, the authors applied first-order forward-Euler scheme, second-order Crank-Nicolson-Adams-Bashfort (CNAB) method, and third-order backward differentiation formula to the diffusion term. For the method with CNAB, the second-order Adams-Bashforth was implemented for both the ODEs and the parabolic PDEs, while Crank-Nicolson was only applied to the PDEs. For all the methods mentioned in this research, the elliptic equation was time-discretized using the forward-Euler.

Most implementations of the semi-implicit methods for the bidomain equations are first- or second-order methods [47]. In Franzonem et al. [48], the authors simulated a full normal heartbeat using the bidomain equations with ODEs. Researchers adaptively changed the time-step for different phases (the upstroke, plateau, and downstroke). The semi-implicit method in this research discretized the diffusion term by the backward-Euler method, and the non-linear reaction term was discretized by the forward-Euler method. In addition, In Whiteley's work [49], the semi-implicit method was applied to update V and ϕ_e, and the backward-Euler was applied for the ODEs. Whiteley [50] also developed a semi-implicit scheme which allows an adaptive numerical solution in both time and space for the bidomain equations. The Crank-Nicolson-forward-Euler method, a second-order semi-implicit method, averages the current and previous V and ϕ_e terms in the parabolic PDE of the bidomain system. According to Ethier et al. [43], this method is considered accurate, as the parabolic part has a truncation error that is second-order in time. However, One disadvantage of this method is that when applying to irregular meshes, it is complicated to calculate the time-step size to satisfy the stability requirement [51].

A different approach was the operator splitting technique, in which case the PDEs and ODEs are decoupled in several ways, and are solved by different time discretization schemes. This approach follows the idea that most ODEs in the bidomain system are non-linear and highly complex, and thus by splitting these equations from the PDEs, the computational cost will be significantly reduced [52]. One operator-splitting method was proposed by Sundnes et al. [53]. In this work, the authors solved the system of ODEs using a forward method first, and used the updated $\mathbf{u}$ for the I_ion term. After that they applied a fully implicit scheme for the coupled PDE system. They mentioned in the paper that this scheme had a much looser stability constraint for the time-step size compared with the previous stated semi-implicit methods.

1.6 Solver Approaches

After the time and spatial discretization, a linear system $Ax = b$ is formed for the bidomain system, which will be shown in chapter 2. In previous research, iterative solvers, geometric multigrid solvers, and algebraic multigrid solvers were applied for solving the bidomain equations.

Iterative method starts at an approximate solution. It applies to the problem repeatedly to reduce the error. Usually the stopping criterion is a value of the norm of the change in residual between two iterations. One method of this type is the conjugate gradient method (CG), which can be applied to large sparse systems that cannot be handled by direct methods. CG changes the original problem of finding the solution of a linear into an optimization problem. A further development of CG is the preconditioned conjugate gradient (PCG), which performs an additional step in CG to make the original problem well-conditioned. The performance of PCG methods depends significantly on the preconditioner applied. Generally, more expensive preconditioners lead to less iterations to achieve the desired accuracy, with higher computational cost per iteration. In Eason et al. [54], diagonal preconditioner was implemented for solving the bidomain system. One preconditioner that is widely applied is the incomplete LU (ILU), which is a sparse approximation of the LU decomposition. With this method, only parts of the original decomposed matrices are retained, and the product of the upper and lower triangular matrix will not be the exact original matrix. The ILU preconditioner was compared with diagonal preconditioner for the bidomain system in a work presented by Potse et al. [23]. According to the results, the diagonal preconditioner was much faster than ILU per iteration. However, the number of convergence iterations for the diagonal preconditioner was greater than that of ILU. Consequently, the total runtime with the diagonal preconditioner was twice greater than that with ILU.

Applying ILU as the preconditioner for the bidomain equations, more fill-in (a less sparse approximation) of the approximation matrix represents the true decomposed matrix better, while resulting in more memory cost [55]. In Gerardo-Giorda et al. [56], applying PCG-ILU to the bidomain equations significantly reduced the number of iterations with a higher level of fill-in. When running in parallel, the matrix is divided among processors, and each portion is solved independently by a processor. Since there is no interaction between the submatrices, the solution in parallel only approximates the true solution. As the amount of fill-in increases, the matrices are denser, and the time for each PCG iteration increases. In this way, the advantage of time-saving of parallelization decreases. A study [41] showed that by applying the block-Jacobi decomposition, in which case ILU is performed for the main diagonal block of the iteration matrix, PCG-ILU with a high level of fill-in was also fast when performing with parallel processors for the bidomain equations. Successive Over-Relaation (SOR) can also be used as a preconditioner for the bidomain system. SOR is a variation of the Gauss-Seidel method. It uses a parameter to overweight the correction term, and leads to a faster convergence. The symmetric SOR (SSOR) combines two SOR sweeps together, in a way that the new iteration matrix is similar to a symmetric matrix. SSOR can be shown to provide a speed-up compared with diagonal preconditioners as a bidomain equations' preconditioner [55]. Weber dos Santos [57] showed that SSOR PCG provide a spped-up over diagonal preconditioning. Other popular preconditioners for the bidomain equations are block preconditioners. With this method, the preconditioning system is partitioned into disjoint sets of equations, and each set can be preconditioned differently. Well-known block preconditioners for the bidomain equations with CG are the block Jacobi and block SSOR methods. In Franzone et al. [48], a parallel solver with the block Jacobi PCG was applied to solve the 3D monodomain and bidomain system. Pennacchino and Simoncini [47] showed that block SSOR PCG substantially reduces the time spent to solve the linear system, without increasing the memory requirements..

For iterative solver for large sparse problem, the generalized minimal residual method (GMRES) is another common choice. In Pathmanathan et al. [20], performance of GMRES preconditioned with block Jacobi was compared with GMRES without preconditioner on the bidomain model. The authors concluded that with block Jacobi, the solving time was noticeably reduced compared with the no preconditioner case. They also considered CG, and mentioned that the difference of the performances of CG and GMRES were not significant. In Gerardo-Giorda et al. [56], in order to reduce CPU time, the researchers applied the flexible GMRES (FGMRES) and solved the preconditioner inaccurately for the monodomain and bidomain equations.

Multigrid methods(MG) are multilevel methods that recursively transfer the residual from the finer grid to the coarser grid in order to handle the low frequency components. After solving the problem directly on the coarsest level, the residual is interpolated back to the original fine grids. There are multiple schemes for the interpolation and extrapolation, for example, the V-cycle, where one starts at the finest level, working to the coarsest, and then work back to the finest. In the numerical application, the iteration at a certain grid level is called the smoothing process, because it can smooth out the high frequency component of the error. The matrix transferring values from a finer level to a coarser level is the restriction matrix, while the prolongation matrix transfer values from the opposite. At each grid level except the coarsest one, iterative methods such as jacobi or Gauss-seidel are applied to reduce error. Previous studies applied MG as direct solver for the bidomain equations and obtained fast convergence [58]. In Sundnes et al. [53], the authors applied MG for bidomain system on a 2D cardiac mesh, and concluded this method to be an efficient solver. In Austin et al. [59], the black-box MG (BBMG), which is a revision of classical MG to deal with discontinuous coefficients, was implemented for test problems with discontinuities arising from inserted plunge electrodes in the heart mesh. The authors concluded that BBMG had a much faster performance compared with classical MG. MG can also be applied as the preconditioner for PCG. In Santos et al [41], the MG preconditioned CG was shown to be suited for quickly and accurately solving the bidomain system, compared with direct MG solver and CG with ILU preconditioner.

Geometric multigrid (GMG) is one type of the MG that the different levels of meshes for the same geometry are created by the user. This method is usually applied for structured methes. To maximize the performance of GMG, it is critical to determine the number of levels. If applying GMG as the bidomain solver in parallel, as the number of processors increases, the number of MG levels should also increase [55]. This is due to the balance of the advantage of parallelism and the computational cost for solving on the coarsest grid for each processor. As the coarsest grid was solved one by one on the processors, less MG levels results in more grids in the coarsest level and a heavier computational cost for solving on the coarsest level. In addition, memory usage should also be considered when deciding the number of grid levels. According to Vigmond et al. [55], the memory demand for GMG is less than direct solver, but is greater than PCG-ILU. Multiple previous works also explored applying GMG as a preconditioner. In Weber Dos Santos et al. [41], for solving the bidomain problem, PCG-GMG showed a much faster speed than that of PCG-ILU on both 2D and 3D electric propagation problems. Another study [60] reported a significant smaller number of iterations using PCG-GMG for the bidomain system than using PCG-ILU. Also, in parallel, as the penalty for domain decomposition of ILU is substantial, the total number of iterations using PCG-ILU suffers a large increase. However, this problem does not apply to PCG-GMG [55]. In Weber Dos Santos et al. [61], PCG-ILU outperformed PCG-ILU with similar memory requirement and about a 20 times faster speed with 8 and 16 processors on a 2D tissue geometry.

Algebraic Multigrid (AMG) is another MG method that can be applied for solving the bidomain equations. The difference between AMG an GMG is that, no information concerning the grid is required for AMG; while the coarser grids are constructed from the finer grids for GMG. By simply examining the matrix structure, the prolongation and restriction operatiors, as well as the coarser representations of the matrix are generated [55]. For unstructured meshes, AMG is useful considering the difficulties of constructing the coarser meshes. This advantage can be used when solving the bidomain equations on the real cardiac meshes, which account for the curved surface of the heart. In Austin et al. [62], the performance of AMG solver and PCG-ILU were compared for solving the elliptic component of the bidomain equations on a 2D cardiac tissue. According to the results, AMG solved the problem much faster than PCG-ILU. However, since the coarser levels components should be setup for AMG, this method required a significant more memory than PCG-ILU. In addition, AMG is commonly applied as a preconditioner for solving the bidomain equations [55]. In Plank et al. [63], the performance of PCG-AMG was compared with PCG-ILU to solve the bidomain system on two 3D rabbit ventricles meshes. The researchers concluded that AMG preconditioner is clearly superior to the ILU preconditioner in terms of the speed of solving. Pennacchio and Simoncini [64] applied PCG-AMG to the block form of the coefficient matrix of the bidomain system and obtained a constant growth of 1 iteration until $168,577$ finite element discretization elements with 12 iterations to converge to a relative tolerance of 10^{-9}. The authors' later work [65] implemented AMG as a preconditioner for FGMRES to the 3D left ventricle mesh, using the block form of the coefficient matrix. The results showed an increasing of 0 or 1 iteration until $1,841,622$ elements with 10 iterations to converge to a relative tolerance of 10^{-6}.

CHAPTER 2

Basic Numerical Methods for the Bidomain Model

2.1 Finite-Difference Spatial Discretization

In the numerical experiment with geometric multigrid methods (GMG) in 2D, I use a second-order finite-difference discretization of the bidomain equations. Consider the grid-aligned case, in which case the conductivities are

$$\sigma_{\mathrm{i}} = \begin{bmatrix} \sigma_{\mathrm{ix}} & 0 \\ 0 & \sigma_{\mathrm{iy}} \end{bmatrix}, \qquad \sigma_{\mathrm{e}} = \begin{bmatrix} \sigma_{\mathrm{ex}} & 0 \\ 0 & \sigma_{\mathrm{ey}} \end{bmatrix}, \qquad \sigma_{\mathrm{b}} \tag{2.1.1}$$

The bidomain with bath system is

$$\begin{aligned} &\chi(C_{\mathrm{m}}\frac{\partial V}{\partial t} + I_{\mathrm{ion}}) - I_{\mathrm{i}}^{(\mathrm{vol})} = \sigma_{\mathrm{ix}}(\frac{\partial^2 \phi_{\mathrm{e}}}{\partial x^2} + \frac{\partial^2 V}{\partial x^2}) + \sigma_{\mathrm{iy}}(\frac{\partial^2 \phi_{\mathrm{e}}}{\partial y^2} + \frac{\partial^2 V}{\partial y^2}), \\ &\sigma_{\mathrm{ix}}\frac{\partial^2 V}{\partial x^2} + \sigma_{\mathrm{iy}}\frac{\partial^2 V}{\partial y^2} + (\sigma_{\mathrm{ix}} + \sigma_{\mathrm{ex}})\frac{\partial^2 \phi_{\mathrm{e}}}{\partial x^2} + (\sigma_{\mathrm{iy}} + \sigma_{\mathrm{ey}})\frac{\partial^2 \phi_{\mathrm{e}}}{\partial y^2} = -I_{\mathrm{total}}^{(\mathrm{vol})}, \\ &\sigma_{\mathrm{b}}\frac{\partial^2 \phi_{\mathrm{b}}}{\partial x^2} + \sigma_{\mathrm{b}}\frac{\partial^2 \phi_{\mathrm{b}}}{\partial y^2} = 0. \end{aligned} \tag{2.1.2}$$

I will describe the non-aligned case in chapter 4. The spatial approximation for this system uses second-order central differences. For example, $\sigma_{\mathrm{ix}}\frac{\partial^2 V_{i,j}}{\partial x^2} + \sigma_{\mathrm{iy}}\frac{\partial^2 V_{i,j}}{\partial y^2}$ is approximated as

$$\sigma_{\mathrm{ix}}\frac{V_{i+1,j} + V_{i-1,j} - 2V_{i,j}}{\Delta x^2} + \sigma_{\mathrm{iy}}\frac{V_{i,j+1} + V_{i,j-1} - 2V_{i,j}}{\Delta y^2} \tag{2.1.3}$$

where Δx^2 and Δy^2 are the node spacing in the x and y directions. In the numerical simulation in this thesis, I use uniform grids, in which case $\Delta x = \Delta y = h$.

2.2 Finite-Element Spatial Discretization

2.2.1 Bidomain Equations without Bath

To spatially discretize the bidomain equations without bath using the finite-element method, the first step is to transform the system into its weak form, which is obtained by multiplying each term in the equations by a test function, together with the boundary conditions. Choosing the test function $\psi \in H^1(\Omega)$ and multiplying it against the strong form of the equations, we have

$$\begin{aligned}&\int_\Omega \chi C_{\mathrm{m}} \frac{\partial V}{\partial t}\psi d\vec{x} - \int_\Omega \nabla\cdot(\sigma_{\mathrm{i}}\nabla(V+\phi_{\mathrm{e}}))\psi d\vec{x}\\&+\int_\Omega (\chi I_{\mathrm{ion}}(\vec{u},V) - I_{\mathrm{i}}^{(\mathrm{vol})})\psi d\vec{x} - \int_{\Gamma_{\mathrm{m}}} \psi\mathbf{n}\cdot(\sigma_{\mathrm{i}}\nabla(V+\phi_{\mathrm{e}}))dS = 0,\\&\int_\Omega \nabla\cdot(\sigma_{\mathrm{i}}\nabla V + (\sigma_{\mathrm{i}}+\sigma_{\mathrm{e}})\nabla\phi_{\mathrm{e}})\psi d\vec{x} + \int_\Omega I_{\mathrm{total}}^{(\mathrm{vol})}\psi d\vec{x} - \int_{\Gamma_{\mathrm{m}}} \psi\mathbf{n}\cdot(\sigma_{\mathrm{i}}\nabla V + (\sigma_{\mathrm{i}}+\sigma_{\mathrm{e}})\phi_{\mathrm{e}})dS = 0.\end{aligned} \tag{2.2.1}$$

To obtain a weak form of the system, we apply the integration by parts, so that

$$\begin{aligned}&\int_\Omega \chi C_{\mathrm{m}} \frac{\partial V}{\partial t}\psi d\vec{x} + \int_\Omega \sigma_{\mathrm{i}}\nabla(V+\phi_{\mathrm{e}})\nabla\psi d\vec{x}\\&+\int_\Omega (\chi I_{\mathrm{ion}}(\vec{u},V) - I_{\mathrm{i}}^{(\mathrm{vol})})\psi d\vec{x} - \int_{\Gamma_{\mathrm{m}}} \mathbf{n}\cdot(\sigma_{\mathrm{i}}\nabla(V+\phi_{\mathrm{e}}))\psi dS = 0,\\&\int_\Omega \sigma_{\mathrm{i}}\nabla V\nabla\psi d\vec{x} + \int_\Omega (\sigma_{\mathrm{i}}+\sigma_{\mathrm{e}})\nabla\phi_{\mathrm{e}}\nabla\psi d\vec{x} + \int_\Omega I_{\mathrm{total}}^{(\mathrm{vol})}\psi d\vec{x} - \int_{\Gamma_{\mathrm{m}}} \mathbf{n}\cdot(\sigma_{\mathrm{i}}\nabla V + (\sigma_{\mathrm{i}}+\sigma_{\mathrm{e}})\phi_{\mathrm{e}})\psi dS = 0.\end{aligned} \tag{2.2.2}$$

According to boundary condition (1.3.21) and (1.3.22) we have

$$\begin{aligned}&\int_{\Gamma_{\mathrm{m}}} \mathbf{n}\cdot(\sigma_{\mathrm{i}}\nabla(V+\phi_{\mathrm{e}}))\psi dS = I_{\mathrm{i}}^{(\mathrm{surf})},\\&\int_{\Gamma_{\mathrm{m}}} \mathbf{n}\cdot(\sigma_{\mathrm{e}}\nabla\phi_{\mathrm{e}})\psi dS = I_{\mathrm{e}}^{(\mathrm{surf})}.\end{aligned} \tag{2.2.3}$$

The weak statement of the bidomain equations is

$$\begin{aligned}&\int_\Omega \chi C_{\mathrm{m}} \frac{\partial V}{\partial t}\psi d\vec{x} - \int_\Omega (\sigma_{\mathrm{i}}\nabla(V+\phi_{\mathrm{e}}))\nabla\psi d\vec{x} + \int_\Omega (\chi I_{\mathrm{ion}}(\vec{u},V) - I_{\mathrm{i}}^{(\mathrm{vol})})\psi d\vec{x} - \int_{\Gamma_{\mathrm{m}}} \psi I_{\mathrm{i}}^{(\mathrm{surf})} dS = 0,\\&\int_\Omega (\sigma_{\mathrm{i}}\nabla V + (\sigma_{\mathrm{i}}+\sigma_{\mathrm{e}})\nabla\phi_{\mathrm{e}})\nabla\psi d\vec{x} + \int_\Omega I_{\mathrm{total}}^{(\mathrm{vol})}\psi d\vec{x} - \int_{\Gamma_{\mathrm{m}}} \psi(I_{\mathrm{e}}^{(\mathrm{surf})} + I_{\mathrm{i}}^{(\mathrm{surf})})dS = 0.\end{aligned} \tag{2.2.4}$$

To obtain a numerical approximation of the weak form, we triangulate the domain into a set of N nodes, and choose a set of basis functions $\psi_1, \psi_2, ..., \psi_N$ spanning a finite-dimensional subspace of Ω. The basis functions used in this study satisfy an interpolation property, so that $\psi_i(x_j) = \delta_{ij}$, where x_j is the j$^{\mathrm{th}}$ node, and the δ_{ij} is the so-called Kronecker delta function. This yields the

Lagrangian finite element basis functions. Let

$$\begin{aligned} V &= \sum V_i \psi_i, \\ \phi_\mathrm{e} &= \sum \Phi_i \psi_i. \end{aligned} \tag{2.2.5}$$

Let K_s be the stiffness matrix for the sth subdomain, $s =$ i, e, or b:

$$\begin{aligned} (K_\mathrm{i})_{jk} &= \int \nabla\psi_j \cdot (\sigma_\mathrm{i} \nabla\psi_k) d\vec{x}, \\ (K_\mathrm{j})_{jk} &= \int \nabla\psi_j \cdot (\sigma_\mathrm{j} \nabla\psi_k) d\vec{x}, \\ (K_\mathrm{b})_{jk} &= \int \nabla\psi_j \cdot (\sigma_\mathrm{b} \nabla\psi_k) d\vec{x}. \end{aligned} \tag{2.2.6}$$

The discretization of the parabolic equation (1.3.18) is

$$\chi C_\mathrm{m} M \dot{\mathbf{V}} + K_\mathrm{i}\mathbf{V} + K_\mathrm{i}\mathbf{\Phi}_\mathrm{e} + \mathbf{I} = 0, \tag{2.2.7}$$

where $V = (V_1, ..., V_N)$ and $\mathbf{\Phi}_\mathrm{e} = (\Phi_1, ..., \Phi_N)$ are vectors of nodal coefficients. The mass matrix M_{ij} is

$$M_{ij} = \int \psi_i \psi_j d\vec{x}, \tag{2.2.8}$$

and

$$I_j = \int_\Omega (I_\mathrm{i}^{(\mathrm{vol})} + \chi I_\mathrm{ion}(\vec{u}, V))\psi_j d\vec{x} - \int_{\Gamma_\mathrm{m}} \psi_j I_\mathrm{i}^{(\mathrm{surf})} dS. \tag{2.2.9}$$

The discretization of the elliptic equation is

$$K_\mathrm{i}\mathbf{V} + (K_\mathrm{i} + K_\mathrm{e})\mathbf{\Phi}_\mathrm{e} + \mathbf{J} = 0, \tag{2.2.10}$$

where

$$\int_\Omega I_\mathrm{total}^{(\mathrm{vol})} \psi_j d\vec{x} - \int_{\Gamma_\mathrm{m}} \psi_j (I_\mathrm{i}^{(\mathrm{surf})} + I_\mathrm{e}^{(\mathrm{surf})}) dS = J_j = 0. \tag{2.2.11}$$

Applying the assumption that $I_\mathrm{e}^{(\mathrm{vol})} = -I_\mathrm{i}^{(\mathrm{vol})}$ and $I_\mathrm{e}^{(\mathrm{surf})} = -I_\mathrm{i}^{(\mathrm{surf})}$, implying an extracellular stimulus at every point where there is an intracellular stimulus with a magnitude equal and opposite.

Since

$$\begin{aligned} I^{(\mathrm{surf})}_{\mathrm{total}} &= I^{(\mathrm{vol})}_{\mathrm{e}} + I^{(\mathrm{vol})}_{\mathrm{i}}, \\ I^{(\mathrm{surf})}_{\mathrm{total}} &= I^{(\mathrm{surf})}_{\mathrm{e}} + I^{(\mathrm{surf})}_{\mathrm{i}}, \end{aligned} \tag{2.2.12}$$

we have

$$\int_{\Omega} I^{(\mathrm{vol})}_{\mathrm{total}} \psi_j d\vec{x} - \int_{\Gamma_{\mathrm{m}}} I^{(\mathrm{surf})}_{\mathrm{total}} \psi_j d\vec{x} = J_j = 0. \tag{2.2.13}$$

2.2.2 Bidomain Equation with Bath

For the bidomain problem with a conductive bath, suppose there are two disjoint domains Ω and Ω_{b} , denoting the muscle and the bath, respectively. We form the first equation of the weak form by multiplying by a test function $\nu \in H^1(\Omega)$ and integrating using the divergence theorem:

$$\int_{\Omega} \chi C_{\mathrm{m}} \frac{\partial V}{\partial t} \nu d\vec{x} - \int_{\Omega} (\sigma_{\mathrm{i}} \nabla (V + \phi_{\mathrm{e}})) \nabla \nu d\vec{x} + \int_{\Omega} (\chi I_{\mathrm{ion}}(\vec{u}, V) - I^{(vol)}_{\mathrm{i}}) \nu d\vec{x} - \int_{\Gamma_{\mathrm{m}}} \nu I^{(\mathrm{surf})}_{\mathrm{i}} dS = 0. \tag{2.2.14}$$

We form the second equation of the weak form by multiplying by a test function $\varphi \in H^1(\Omega)$, and integrating using the divergence theorem:

$$\begin{aligned} &\int_{\Omega} (\sigma_{\mathrm{i}} \nabla V + (\sigma_{\mathrm{i}} + \sigma_{\mathrm{e}}) \nabla \phi_e) \nabla \varphi d\vec{x} + \int_{\Omega} I^{(\mathrm{vol})}_{\mathrm{total}} \varphi d\vec{x} - \int_{\Gamma_{\mathrm{m}}} \varphi (I^{(\mathrm{surf})}_{\mathrm{e}} + I^{(\mathrm{surf})}_{\mathrm{i}}) dS \\ &- \int_{\Gamma_{\mathrm{m}}} \mathbf{n} \cdot (\sigma_{\mathrm{e}} \nabla \phi_{\mathrm{e}}) \varphi dS - \int_{\Gamma_{\mathrm{i}}} \varphi (I^{(\mathrm{surf})}_{\mathrm{e}} + I^{(\mathrm{surf})}_{\mathrm{i}}) dS - \int_{\Gamma_{\mathrm{i}}} \mathbf{n} \cdot (\sigma_{\mathrm{e}} \nabla \phi_{\mathrm{e}}) \varphi dS = 0. \end{aligned} \tag{2.2.15}$$

Applying the assumption that $I^{(\mathrm{vol})}_{\mathrm{e}} = -I^{(\mathrm{vol})}_{\mathrm{i}}$ and $I^{(\mathrm{surf})}_{\mathrm{e}} = -I^{(\mathrm{surf})}_{\mathrm{i}}$ on Ω and Γ_{m}, respectively, we have

$$\int_{\Omega} (\sigma_{\mathrm{i}} \nabla V + (\sigma_{\mathrm{i}} + \sigma_{\mathrm{e}}) \nabla \phi_e) \nabla \varphi d\vec{x} - \int_{\Gamma_{\mathrm{m}}} \mathbf{n} \cdot (\sigma_{\mathrm{e}} \nabla \phi_{\mathrm{e}}) \varphi dS - \int_{\Gamma_{\mathrm{i}}} \mathbf{n} \cdot (\sigma_{\mathrm{e}} \nabla \phi_{\mathrm{e}}) \varphi dS = 0. \tag{2.2.16}$$

Recall equation (1.3.27) for ϕ_{b}. To transfer this equation into its weak form, I multiply equation (1.3.27) and the boundary condition (1.3.33) by a test function $\omega \in H^1(\Omega_{\mathrm{b}})$,

$$\int_{\Omega_{\mathrm{b}}} \nabla \cdot (\sigma_{\mathrm{b}} \nabla \phi_{\mathrm{b}}) \omega d\vec{x} - \int_{\Gamma_{\mathrm{b}}} \mathbf{n} \cdot (\sigma_{\mathrm{b}} \nabla \phi_{\mathrm{b}}) \omega dS - \int_{\Gamma_{\mathrm{i}}} \mathbf{n} \cdot (\sigma_{\mathrm{b}} \nabla \phi_{\mathrm{b}}) \omega dS = 0. \tag{2.2.17}$$

To reduce the second-order derivatives in the system, we apply the integration by parts, such that

$$\int_{\Omega_\mathrm{b}} \sigma_\mathrm{b}\nabla\sigma_\mathrm{b}\nabla\omega d\vec{x} - \int_{\Gamma_\mathrm{b}} \mathbf{n}\cdot(\sigma_\mathrm{b}\nabla\phi_\mathrm{b})\omega dS - \int_{\Gamma_\mathrm{i}} \mathbf{n}\cdot(\sigma_\mathrm{b}\nabla\phi_\mathrm{b})\omega dS = 0. \tag{2.2.18}$$

According to boundary condition (1.3.33), the weak formula of the bath part is

$$\int_{\Omega_\mathrm{b}} \sigma_\mathrm{b}\nabla\sigma_\mathrm{b}\nabla\omega d\vec{x} - \int_{\Gamma_\mathrm{b}} \omega I_\mathrm{b}^{(\mathrm{surf})} dS - \int_{\Gamma_\mathrm{i}} \mathbf{n}\cdot(\sigma_\mathrm{b}\nabla\phi_\mathrm{b})\omega dS = 0. \tag{2.2.19}$$

Taking $\omega = \psi$ on $\partial\Gamma_\mathrm{i}$, and summing equation (2.2.16) and (2.2.19), we get

$$\begin{aligned}&\int_\Omega(\sigma_\mathrm{i}\nabla V + (\sigma_\mathrm{i}+\sigma_\mathrm{e})\nabla\phi_\mathrm{e})\nabla\varphi d\vec{x} + \int_{\Omega_\mathrm{b}}\sigma_\mathrm{b}\nabla\phi_\mathrm{b}\nabla\omega d\vec{x} - \int_{\Gamma_\mathrm{b}}\omega I_\mathrm{b}^{(\mathrm{surf})}dS \\ &-(\int_{\Gamma_\mathrm{i}}\mathbf{n}\cdot(\sigma_\mathrm{b}\nabla\phi_\mathrm{b})\omega dS + \int_{\Gamma_\mathrm{i}}\mathbf{n}\cdot(\sigma_\mathrm{e}\nabla\phi_\mathrm{e})\omega dS) = 0.\end{aligned} \tag{2.2.20}$$

Due to the boundary condition (1.3.34), we have

$$\int_\Omega(\sigma_\mathrm{i}\nabla V + (\sigma_\mathrm{i}+\sigma_\mathrm{e})\nabla\phi_\mathrm{e})\nabla\varphi d\vec{x} + \int_{\Omega_\mathrm{b}}\sigma_\mathrm{b}\nabla\phi_\mathrm{b}\nabla\omega d\vec{x} - \int_{\Gamma_\mathrm{b}}\omega I_\mathrm{b}^{(\mathrm{surf})}dS = 0. \tag{2.2.21}$$

To describe the finite-element method for the bidomain with bath case, assume there are K nodes in Ω, $N-K$ nodes in Γ_i, and M nodes in Ω_b. Suppose $x_1, ..., x_K$ are in Ω, $x_{K+1}, .., x_N$ are in Γ_i, and $x_{N+1}, ..., x_{N+M}$ are in Ω_b. The basis functions become

$$\psi_1, ..., \psi_K, \psi_{K+1}, ..., \psi_N, \psi_{N+1}, ..., \psi_{N+M}, \tag{2.2.22}$$

where $\psi_1, ..., \psi_K$ are zero in Ω_b, and $\psi_{N+1}, ..., \psi_{N+M}$ are zero in Ω.

For the discretization of equation (2.1.14), let $V = \sum_{j=1}^N V_j\psi_j$, and $\phi_\mathrm{e} = \sum_{j=1}^N \Phi_j\psi_j$, indicating that V and ϕ_e are considered only in Ω for this equation. Let $\mathbf{V} = (V_1, ..., V_N)$ and $\mathbf{\Phi}_\mathrm{e} = (\Phi_1, ..., \Phi_N)$. Letting $\psi = \psi_j, j = 1, ..., N$ in equation (2.1.14), the discretized first equation is

$$\chi C_\mathrm{m} M\dot{\mathbf{V}} + K_\mathrm{i}\mathbf{V} + K_\mathrm{i}\mathbf{\Phi}_\mathrm{e} + \mathbf{I} = 0, \tag{2.2.23}$$

where

$$I_j = \int_\Omega (I_{\rm i}^{(\rm vol)} + \chi I_{\rm ion}(\vec{u}, V))\psi_j d\vec{x} - \int_{\Gamma_{\rm m}} \psi_j I_{\rm i}^{(\rm surf)} dS, \tag{2.2.24}$$

and M is the $N \times N$ mass stiffness matrix, and K_i is the $N \times N$ stiffness matrix using $\sigma_{\rm i}$.

For the discretization of (2.2.21), we consider V and $\phi_{\rm e}$ in the tissue, and $\phi_{\rm b}$ in the bath, let $\varphi = \psi_1, ..., \psi_N$ and $\omega = \psi_{K+1}, ..., \psi_{N+M}$. The discretization is

$$\begin{aligned}&\sum_{k=1}^{N}(V_k \int_\Omega (\sigma_{\rm i}\nabla\psi_k)\cdot\nabla\psi_j d\vec{x} + \Phi_k \int_\Omega ((\sigma_{\rm i}+\sigma_{\rm e})\nabla\psi_k)\cdot\nabla\psi_j d\vec{x}) + \sum_{k=K+1}^{N+M}\Phi_k(\int_{\Omega_{\rm b}} (\sigma_{\rm b}\nabla\psi_k)\cdot\nabla\psi_j d\vec{x}\\ &- \int_{\Gamma_{\rm b}} I_{\rm b}^{(\rm surf)}\psi_j dS) = 0,\end{aligned} \tag{2.2.25}$$

Written in matrix form, and let $\mathbf{\Phi}_{\rm b} = (\Phi_{K+1}, ..., \Phi_M)$, the discretization is

$$\begin{aligned}&K_{\rm i}\mathbf{V} + (K_{\rm i} + K_{\rm e})\mathbf{\Phi}_{\rm e} = 0,\\ &K_{\rm b}\Phi_b - \mathbf{T} = 0.\end{aligned} \tag{2.2.26}$$

where

$$\mathbf{T}_j = \int_{\Gamma_{\rm b}} I_{\rm b}^{(\rm surf)}\psi_j dS. \tag{2.2.27}$$

2.3 Semi-Implicit Time Stepping Method

In the numerical experiments, we use the semi-implicit time discretization. Given the time-step size Δt, we use the notation

$$\begin{cases} V^n(\mathbf{x}) = V(\mathbf{x}, n\Delta t),\\ \phi_{\rm e}^n(\mathbf{x}) = \phi_{\rm e}(\mathbf{x}, n\Delta t),\\ \vec{u}^n(\mathbf{x}) = \vec{u}(\mathbf{x}, n\Delta t),\\ \phi_{\rm b}^n(\mathbf{x}) = \phi_{\rm b}(\mathbf{x}, n\Delta t).\end{cases} \tag{2.3.1}$$

In equation (2.1.23), we discretize $\chi C_{\rm m} M\dot{\mathbf{V}}$ as

$$\frac{\chi C_{\rm m} M(\mathbf{V}^n - \mathbf{V}^{n-1})}{\Delta t}. \tag{2.3.2}$$

In the time discretization, the diffusion term $K_\text{i}\mathbf{V} + K_\text{i}\mathbf{\Phi}_\text{e}$ is treated implicitly as

$$K_\text{i}\mathbf{V}^n + K_\text{i}\mathbf{\Phi}_\text{e}^n. \tag{2.3.3}$$

We treat the reaction term I_ion in equation (2.1.23) explicitly as I_ion^{n-1}.

We discretize equations in (2.1.26) as

$$\begin{aligned} &K_\text{i}\mathbf{V}^n + (K_\text{i} + K_\text{e})\mathbf{\Phi}_\text{e}^n = 0, \\ &K_\text{b}\mathbf{\Phi}_\text{b}^n - \mathbf{T} = 0. \end{aligned} \tag{2.3.4}$$

The discretized bidomain system with bath in matrix form is

$$\begin{bmatrix} \frac{\chi C_\text{m}}{\Delta t} M + K_\text{i} & K_\text{i} & 0 \\ K_\text{i} & \kappa_{(1,1)} & \kappa_{(1,2)} \\ 0 & \kappa_{(2,1)} & \kappa_{(2,2)} \end{bmatrix} \begin{bmatrix} \mathbf{V}^n \\ \mathbf{\Phi}^n \\ \mathbf{\Phi}_\text{b}^n \end{bmatrix} = \begin{bmatrix} \frac{\chi C_\text{m}}{\Delta t} M\mathbf{V}^{n-1} - \mathbf{I} \\ 0 \\ \mathbf{T} \end{bmatrix} \tag{2.3.5}$$

where the entries in the 2 by 2 matrix

$$\begin{bmatrix} \kappa_{(1,1)} & \kappa_{(1,2)} \\ \kappa_{(2,1)} & \kappa_{(2,2)} \end{bmatrix} \tag{2.3.6}$$

are

$$\kappa_{kj} = \int_\Omega ((\sigma_\text{i} + \sigma_\text{e})\nabla\phi_k) \cdot \nabla\phi_j) d\vec{x} + \int_{\Omega_\text{b}} (\sigma_\text{b}\nabla\phi_k) \cdot \nabla\phi_j) d\vec{x}. \tag{2.3.7}$$

CHAPTER 3

Multigrid for Elliptic PDEs

3.1 Classical Iterative Methods for Linear Systems

Iterative methods use an initial guess to generate a sequence of approximated solutions, and each of the approximation is generated from the previous ones. For a linear system $A\mathbf{x} = \mathbf{b}$, basic iterative methods usually split the A matrix into a sum of several matrices. For example, if splitting A into $Q - (Q - A)$, we have

$$Q\mathbf{x} = \mathbf{b} + (Q - A)\mathbf{x}. \tag{3.1.1}$$

This splitting can motivate an iterative process,

$$Q\mathbf{x}^m = \mathbf{b} + (Q - A)\mathbf{x}^{m-1}. \tag{3.1.2}$$

We can initiate the iterative process with the initial value $\mathbf{x}^0$. We hope to be able to choose the matrix Q so that the iterative process will converge to the true solution $\mathbf{x}$ with a small number of iterations. The following three schemes are examples of commonly used classical iterative methods:

- Richardson iteration: $Q = I$

$$\mathbf{x}^m = \mathbf{b} + (I - A)\mathbf{x}^{m-1}. \tag{3.1.3}$$

- Jacobi iteration: $Q = \text{diag}(A)$, denoted as D

$$\mathbf{x}^m = D^{-1}(\mathbf{b} - R\mathbf{x}^{m-1}), \tag{3.1.4}$$

  where $R = A - D$.

- Gauss-Seidel: Q is the strictly lower triangular part of A, denoted as L

$$\mathbf{x}^m = (D + L)^{-1}(\mathbf{b} - U\mathbf{x}^{m-1}), \tag{3.1.5}$$

where U is the strictly upper triangular part of A, with $A = L + D + U$.

We can rearrange equation (3.1.2) as

$$\begin{aligned} \mathbf{x}^m &= Q^{-1}(Q - A)\mathbf{x}^{m-1} + Q^{-1}\mathbf{b} \\ &= (I - Q^{-1}A)\mathbf{x}^{\text{m-1}} + Q^{-1}\mathbf{b}. \end{aligned} \tag{3.1.6}$$

For convergence analysis, subtract $\mathbf{x} = (I - Q^{-1}A)\mathbf{x} + Q^{-1}\mathbf{b}$ from $\mathbf{x}^m = (I - Q^{-1}A)\mathbf{x}^{m-1} + Q^{-1}\mathbf{b}$, we have

$$(\mathbf{x}^m - \mathbf{x}) = (I - Q^{-1}A)(\mathbf{x}^{m-1} - \mathbf{x}). \tag{3.1.7}$$

Take the norms on both sides:

$$\begin{aligned} ||(\mathbf{x}^m - \mathbf{x})|| &= ||(I - Q^{-1}A)(\mathbf{x}^{m-1} - \mathbf{x})|| \\ &\leq ||(I - Q^{-1}A)||||(\mathbf{x}^{m-1} - \mathbf{x})|| \\ &\leq ||(I - Q^{-1}A)||^2||(\mathbf{x}^{m-2} - \mathbf{x})|| \\ &\vdots \\ &\leq ||(I - Q^{-1}A)||^m||(\mathbf{x}^0 - \mathbf{x})||. \end{aligned} \tag{3.1.8}$$

Thus if

$$\lim_{m\to\infty} ||(I - Q^{-1}A)||^m = 0. \tag{3.1.9}$$

the iterative process converges. Since for any matrix M,

$$\rho(M) \leq ||M||. \tag{3.1.10}$$

For the natural matrix norm $|| \cdot ||$, where

$$\rho(M) = \max(|\lambda_1|, ..., |\lambda_n|). \tag{3.1.11}$$

If

$$\rho(I - Q^{-1}A) < 1, \tag{3.1.12}$$

the iterative process converges.

3.1.1 Basic Idea for Multigrid Methods

Suppose for an one dimensional elliptic equation with Dirichlet boundary

$$-u'' = f, \quad x \in (0,1) \quad u(0) = u(1) = 0. \tag{3.1.13}$$

If applying the second-order finite difference discretization

$$u''(x_i) \approx \frac{1}{h^2}(u_{i+1} - 2u_i + u_{i-1}), \tag{3.1.14}$$

the linear system $A\mathbf{u} = \mathbf{b}$ will be formed, where

$$A = \frac{1}{h^2}\begin{bmatrix} -2 & 1 & 0 & ... \\ -1 & -2 & 1 & ... \\ & \ddots & & \\ ... & 0 & 1 & -2 \end{bmatrix}, \quad \mathbf{b} = (b_1, ..., b_i), \quad b_i = f(x_i). \tag{3.1.15}$$

To calculate the eigenvalues for A, we are solving

$$-\frac{(u_{i+1} - 2u_i + u_{i-1})}{h^2} = \lambda u_i, \quad i = 1, ..., n, \quad u_0 = u_{n+1} = 0. \tag{3.1.16}$$

Rearranging terms

$$u_{i+1} = (2 - h^2\lambda)u_i - u_{i-1}. \tag{3.1.17}$$

which is similar to the Chebyshev polynomials of the second kind.

$$\begin{aligned} U_0(x) &= 1, \\ U_1(x) &= 2x, \\ U_{n+1}(x) &= 2xU_n(x) - U_{n-1}(x). \end{aligned} \tag{3.1.18}$$

In order to transfer (3.1.17) to match the form of Chebyshev polynomials of the second kind, let $2\alpha = (2 - h^2\lambda)$. Assume $u_1 \neq 0$, and scale the associated eigenvector so that $u_1 = 1$. The recurrence relation is

$$\begin{aligned} &u_0 = 0, \\ &u_1 = 1, \\ &u_{i+1} = 2\alpha u_i + u_{i-1}. \end{aligned} \tag{3.1.19}$$

Consider U_k as the kth Chebyshev polynomial of the second kind, and let α satisfy

$$\begin{aligned} &u_{k+1} = U_k(\alpha), \\ &u_1 = U_0(\alpha) = 1, \\ &u_2 = U_1(\alpha) = 2\alpha u_1 - u_0 = 2\alpha. \end{aligned} \tag{3.1.20}$$

which match the form of Chebyshev polynomial of the second kind. As $u_{n+1} = 0$,

$$U_n(\alpha) = 0. \tag{3.1.21}$$

Thus α will be the root of the n^th Chebyshev polynomial of the second kind. The roots are

$$\alpha_k = \cos\left(\frac{k\pi}{n+1}\right). \tag{3.1.22}$$

Plugging these into equation $2\alpha = (2 - h^2\lambda)$

$$2\cos\left(\frac{k\pi}{n+1}\right) = -h^2\lambda_k + 2. \tag{3.1.23}$$

Solving for λ_k

$$\lambda_k = \frac{2}{h^2}\left[1 - \cos\left(\frac{k\pi}{n+1}\right)\right]. \tag{3.1.24}$$

Using the triggonometric formula, this can be simplified to

$$\lambda_k = \frac{2}{h^2}\sin^2\left(\frac{k\pi}{2(n+1)}\right). \tag{3.1.25}$$

In equation (3.1.25), the integer k is the frequency. If k in the range $\left[1, \frac{1}{2}(n+1)\right)$, it is a low frequency; if it is in the range $\left[\frac{1}{2}(n+1), n\right]$, it is a high frequency. To show that classic iteration methods smooth the high frequency part of the error quickly, while leaving the low frequency part slowly, consider the example of Richardson iteration,

$$\mathbf{u}^{k+1} = \mathbf{u}^k + \omega(\mathbf{b} - A\mathbf{u}^k). \tag{3.1.26}$$

The error equation is

$$\begin{aligned} &\mathbf{u} - \mathbf{u}^{k+1} \\ &= \mathbf{u} - \mathbf{u}^k - \omega(A\mathbf{u} - A\mathbf{u}^k) \\ &= (I - \omega A)(\mathbf{u} - \mathbf{u}^k). \end{aligned} \tag{3.1.27}$$

Therefore

$$\begin{aligned} &\mathbf{u} - \mathbf{u}^m = (I - \omega A)^m(\mathbf{u} - \mathbf{u}^0), \\ &\mathbf{e}^m = (I - \omega A)^m \mathbf{e}^0. \end{aligned} \tag{3.1.28}$$

To show the error decay rate for different frequency, expand $\mathbf{e}^0 = \mathbf{u} - \mathbf{u}^0$

$$\mathbf{e}^0 = \sum_{k=1}^{N} \beta_k \xi^k, \tag{3.1.29}$$

where ξ are the eigenvectors of A. Then

$$\mathbf{e}^m = (I - \omega A)^m \mathbf{e}^0 = \sum_{k=1}^{N} \beta_{m,k} \xi^k, \tag{3.1.30}$$

where

$$\beta_{m,k} = (1 - \omega\lambda_k)^m \beta_k. \tag{3.1.31}$$

For the k$^{\text{th}}$ component, the coefficient decays with rate

$$|\beta_{m,k}| \leq |1 - \omega\lambda_k|^m |\beta_k|. \tag{3.1.32}$$

Plugging in values for λ_k,

$$|1 - \omega\lambda_k| = 1 - \frac{2\omega}{h^2}\left(1 - \cos\left(\frac{k\pi}{n+1}\right)\right). \tag{3.1.33}$$

For simplicity, choose $\omega = \frac{h^2}{4}$

$$|1 - \omega\lambda_k| = \frac{1}{2}\left(1 + \cos\left(\frac{k\pi}{n+1}\right)\right). \tag{3.1.34}$$

According to (3.1.34), For the low frequency case, $|1 - \omega\lambda_k| \in [\frac{1}{2}, 1)$; while for the high frequency case, $|1 - \omega\lambda_k| \in (0, \frac{1}{2})$. According to this example, classical iterative methods, such as Richardson iteration, only damp the high frequency components fast.

3.2 Multigrid Methods

The key ideas of multigrid methods (MG) are that, simple iterative methods often can be constructed to rapidly eliminate high frequency errors, and that the low-frequency component of errors on a fine mesh become high-frequency component of errors on a coarser mesh. At least for idealized problems, through transferring the error from a fine grid to a coarse grid, we can smooth out the original low-frequency component by classical iterative methods.

Two types of MG will be compared in this thesis, GMG and AMG. There are two types of grid transferring in MG: restriction and prolongation. The restriction operator transfers values from fine grids to coarse grids, while the prolongation operator extends data from coarse grids to fine grids. The difference between AMG and GMG are the grid transferring process. For GMG, the transferring operators are based on the geometric information (figure (3.1)).

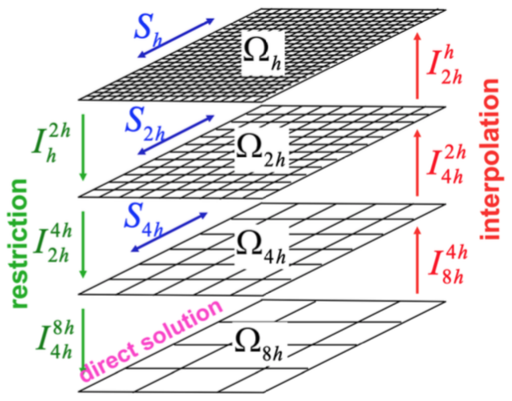

Figure 3.1: For GMG, we use restriction operators to transfer the original problem to a coarser mesh. After solving on the coarser mesh, we transfer the residual back to the finer mesh using the interpolation operator. *http://feflow.info/uploads/media/Stueben.pdf*

For AMG we completely ignore the geometric information when performing the grid transferring (figure (3.2)). AMG extract all needed information from the system matrix. When the original A matrix is given, the linear systems on the "coarser" levels are automatically constructed, without the knowledge of the mesh.

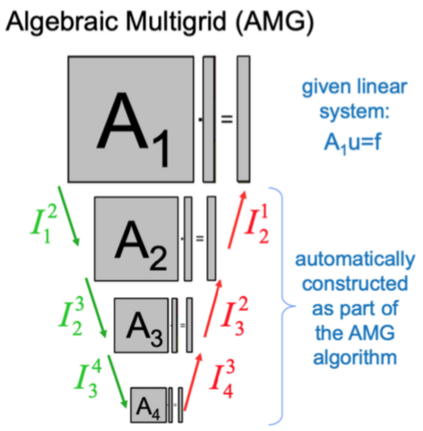
Algebraic Multigrid (AMG)
A_1
given linear system: $A_1u=f$
I_1^2
A_2
I_2^1
I_2^3
A_3
I_3^2
I_3^4
A_4
I_4^3
automatically constructed as part of the AMG algorithm

Figure 3.2: For AMG, we ignore the geometric information when constructing the linear systems on the "coarser" levels. *http://feflow.info/uploads/media/Stueben.pdf*

3.2.1 Operator Construction

According to Briggs et al. [1], it is almost universal that the coarse grid has twice the grid spacing of the next finest grid, since using grid spacings with a ratio more than 2 provides no advantage. For most MG practice, linear interpolation, which is the simplest of the interpolation methods, is applied. Denoting the linear interpolation operator as I_{2h}^h. For one dimensional problems, it takes coarse-grid vectors and generates fine-grid vectors according to $I_{2h}^h\mathbf{v}^{2h} = \mathbf{v}^h$, where

$$\begin{aligned} v_{2j}^h &= v_j^{2h}, \\ v_{2j+1}^h &= \frac{1}{2}(v_j^{2h} + v_{j+1}^{2h}), \quad 0 \le j \le \frac{n}{2} - 1. \end{aligned} \tag{3.2.1}$$

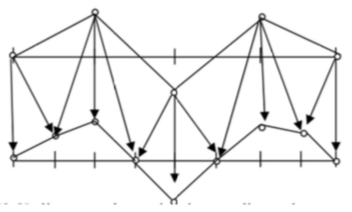

Figure 3.3: Prolongation in 1D *https://scholar.najah.edu/sites/default/files/all-thesis/*

In this case, at even-numbered points, the values are transferred directly from Ω^{2h} to Ω^h; while at odd-numbered points, the values are the average of the adjacent coarse-grid values [1]. Writing in matrix form, it becomes

$$I_{2h}^h \mathbf{v}^{2h} = \frac{1}{2}\begin{bmatrix} 1 & & \\ 2 & & \\ 1 & 1 & \\ & 2 & \\ & 1 & 1 \\ & & 2 \\ & & 1 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix}_{2h} = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \\ v_7 \end{bmatrix}_h = \mathbf{v}^h. \tag{3.2.2}$$

For two-dimensional problems, the interpolation operator can be written as

$$\begin{cases} v_{2i,2j}^h = v_{i,j}^{2h}, \quad \text{for coarse grid points} \\ v_{2i+1,2j}^h = \frac{1}{2}(v_{i,j}^{2h} + v_{i+1,j}^{2h}), \quad \text{for points between the two coarse grid points horizontally} \\ v_{2i,2j+1}^h = \frac{1}{2}(v_{i,j}^{2h} + v_{i,j+1}^{2h}), \quad \text{for points between the two coarse grid points vertically} \\ v_{2i+1,2j+1}^h = \frac{1}{4}(v_{i,j}^{2h} + v_{i,j+1}^{2h} + v_{i,j+1}^{2h} + v_{i+1,j+1}^{2h}), \quad \text{for points on the vertex of the square} \end{cases} \tag{3.2.3}$$

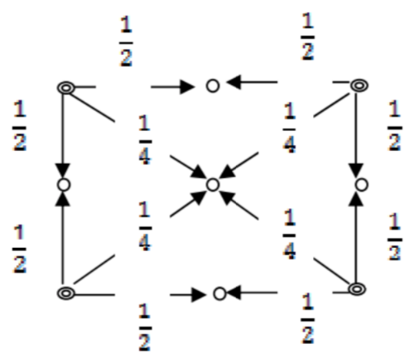

Figure 3.4: Prolongation in 2D *https://scholar.najah.edu/sites/default/files/all-thesis/*

According to Briggs et al. [1], if the real error is smooth, assuming the approximation is exact at the coarse level, when this approximation is interpolated to the fine level, the interpolation is smooth, and thus we should obtain a good approximation of the error in the fine level (figure (3.5a)). However, if the real error is oscillatory, even the exact approximation of the error at the coarse level produces a not accurate approximation of the error in the fine level after the interpolation (figure (3.5b)).

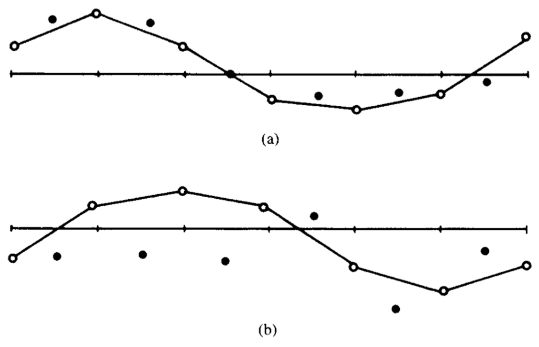

Figure 3.5: (a) If the error (dots in the figure) on the coarse level is smooth, the interpolation (line connecting the dots) should give a a good approximation of the error on the fine level. (b) If the error (dots in the figure) on the coarse level is oscillatory, the interpolation (line connecting the dots) cannot give a a good approximation of the error on the fine level [1].

To transfer from the fine level to the coarse level, we apply the restriction operators (I_h^{2h}).

One common restriction operator is the full weighting operator, which is defined by

$$v_j^{2h} = \frac{1}{4}(v_{2j-1}^h + 2v_{2j}^h + v_{2j+1}^h), \quad 1 \le j \le \frac{n}{2} - 1. \tag{3.2.4}$$

Writing in matrix form, it becomes

$$I_h^{2h}\mathbf{v}^h = \frac{1}{4}\begin{bmatrix} 1 & 2 & 1 & & & & \\ & & 1 & 2 & 1 & & \\ & & & & 1 & 2 & 1 \end{bmatrix}\begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \\ v_7 \end{bmatrix}_h = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix}_{2h} = \mathbf{v}^{2h}. \tag{3.2.5}$$

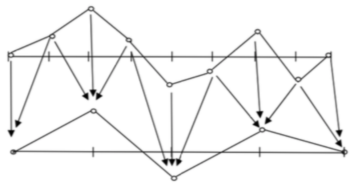

Figure 3.6: Restriction in 1D *https://scholar.najah.edu/sites/default/files/all-thesis/*

According to Briggs et al. [1], if choosing the full weighting restriction operator, we have

$$I_{2h}^h = c(I_{2h}^h)^{\mathrm{T}}, \tag{3.2.6}$$

for a scalar c, so that the prolongation and restriction operators are rescaled to be the transposes of each other.

In 2D the full weighting restriction operator can be written as

$$
\begin{aligned}
v_{ij}^{2h} &= \frac{1}{16}[v_{2i-1,2j-1}^h + v_{2i-1,2j+1}^h + v_{2i+1,2j-1}^h + v_{2i+1,2j+1}^h \\
&+ 2(v_{2i,2j-1}^h + v_{2i,2j+1}^h + v_{2i-1,2j}^h + v_{2i+1,2j}^h) \\
&+ 4v_{2i,2j}^h], \quad 1 \le i, j \le \frac{n}{2} - 1.
\end{aligned} \tag{3.2.7}
$$

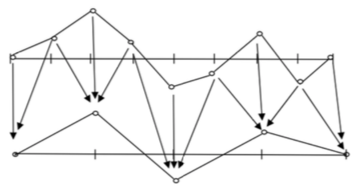

Figure 3.7: Restriction in 2D *https://scholar.najah.edu/sites/default/files/all-thesis/*

3.2.2 Two-grid Simple Case

MG begins with a smoothing process for the error. After the smoothing step, we reduce the high frequency components of error. To reduce the low frequency components, we apply a coarse-grid correction procedure. Once the problem on the coarser grid is solved, we interpolate the solution back to the fine grid, in order to correct the fine grid approximation of the low frequency errors.

The two-grid GMG only involves one fine level and one coarse level. Suppose the elliptic PDE after discretization on uniform grid with mesh size h is

$$A_h \mathbf{u}_h = \mathbf{f}_h. \tag{3.2.8}$$

Let $\mathbf{u}_h^k$ be the approximate solution after k relaxation sweeps, the error will be

$$\mathbf{e}_h^k = \mathbf{u}_h - \mathbf{u}_h^k. \tag{3.2.9}$$

The steps for a two-grid GMG are

- Approximate solution $\mathbf{u}_h^k$ by a classical iterative method
- Compute the residual $\mathbf{r}_h^k = \mathbf{f}_h - A_h \mathbf{u}_h^k$

- Transfer the residual to the coarse grid $\mathbf{r}_H^k = I_h^h \mathbf{r}_h^k$

- Solve the residual equation $A_H \mathbf{e}_H^k = \mathbf{r}_H^k$

- Transfer the error to the fine grid $\mathbf{e}_h^k = I_H^h \mathbf{e}_H^k$

- Compute the new approximation $\mathbf{u}_h^{k+1} = \mathbf{u}_h^k + \mathbf{e}_h^k$

In practice, we usually apply 1 to 3 relaxation sweeps before transferring to the next level. Relaxation on the fine grid eliminates the oscillatory components of the error, leaving the error to be smooth, which makes the interpolation work well.

3.2.3 GMG V-cycle

We can extend the two grid V-cycle to more levels. For example, in the V-cycle in figure 3.8, the GMG algorithm goes down to coarser grids $(2h, 4h, 8h)$, and then back to $(4h, 2h, h)$. The algorithm is the recursive application of the following steps on each level:

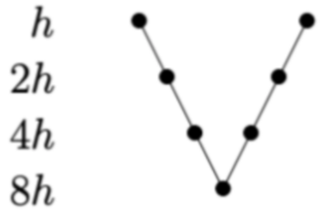

Figure 3.8: Multigrid V-cycle on four levels. *https://www.researchgate.net/figure/Schedule-of-grids-for-a-V-cycle-b-W-cycle-and-c-FMG-scheme-all-on-four-levels_fig10_220690328*

$$\mathbf{u}^h \leftarrow M^h(\mathbf{u}^h, \mathbf{f}^h)$$

- Pre-smoothing: apply the smoother k times to $A^h \mathbf{u}^h = \mathbf{f}^h$ with initial guess $\mathbf{u}_0^h$

- If Ω_h is the coarsest grid, then solve the problem

  else

  - Restrict the residual to the next coarser grid: $\mathbf{r}^{2h} \leftarrow R_h^{2h}(\mathbf{f}^h - A^h \mathbf{u}^h)$

  - On the coarser gird, set initial guess $\mathbf{u}_0^{2h} = 0$

– Apply the V-cycle for the next coarser grid

$$\mathbf{u}^{2h} \leftarrow M^{2h}(\mathbf{u}^{2h}, \mathbf{f}^{2h}) \tag{3.2.10}$$

– Update the solution with prolongation matrix: $\mathbf{u}^h \leftarrow \mathbf{u}^h + P_{2h}^h \mathbf{u}^{2h}$

– Post smoothing: apply the smoother k_1 times to $A^h \mathbf{u}^h = \mathbf{f}^h$ with initial guess $\mathbf{u}_0^h$

3.3 Algebraic Multigrid for Elliptic Equations

For all MG algorithms, the fundamental components are the sequence of grids, the intergrid transferring operators, the relaxation scheme, and the solver on the coarsest grid. Different from GMG, Algebraic Multigrid (AMG) requires no explicit knowledge of the problem geometry. AMG determines coarse grids and inter-grid transfer operators based on the entries of the A matrix only.

3.3.1 Basic Idea

Suppose the one dimensional elliptic equation

$$-u'' = 0, \quad x \in \Omega \tag{3.3.1}$$

is discretized on mesh (figure 3.9) with piecewise-linear finite elements.

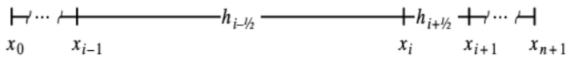

Figure 3.9: Discretization mesh for equation (3.3.1) *http://citeseerx.ist.psu.edu/viewdoc/*

Choose the test function ψ and multiplying it against the strong form of the equations, we have

$$\int_{x_0}^{x_{n+1}} u'' \psi dx = 0. \tag{3.3.2}$$

Applying integration by parts, we have

$$\int_{x_0}^{x_{n+1}} u' \psi' dx = 0. \tag{3.3.3}$$

Choose a set of basis function $\psi_1, \psi_2, ..., \psi_{\mathrm{N}}$ (figure (3.10)), let

$$u = \sum_{i=0}^{n+1} c_i \psi_i. \tag{3.3.4}$$

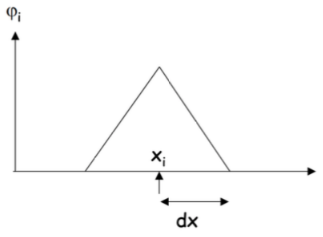

Figure 3.10: Node grid x_i in 1D and the basis functions ψ_i *https://www.geophysik.uni-muenchen.de*

we have

$$\begin{aligned} &\int_{x_0}^{x_{n+1}} \left(\sum_{i=0}^{n+1} c_i \psi_i \right)' {\psi_k}' dx = 0, \\ \Rightarrow \quad &\sum_{i=0}^{n+1} c_i \int_{x_0}^{x_{n+1}} \psi_i' \psi_k' dx = 0. \end{aligned} \tag{3.3.5}$$

The entry A_{ik} in the discretization matrix is

$$A_{ik} = \int_{x_0}^{x_{n+1}} \psi_i' \psi_k' dx. \tag{3.3.6}$$

where

$$\begin{aligned} A_{ii} &= \int_{x_0}^{x_{n+1}} \psi_i' \psi_i' dx, \\ &= \int_{x_i - h_{i-\frac{1}{2}}}^{x_i} \psi_i' \psi_i' dx + \int_{x_i}^{x_i + h_{i+\frac{1}{2}}} \psi_i' \psi_i' dx, \\ &= \int_{x_i - h_{i-\frac{1}{2}}}^{x_i} \frac{1}{h_{i-\frac{1}{2}}} \frac{1}{h_{i-\frac{1}{2}}} dx + \int_{x_i}^{x_i + h_{i+\frac{1}{2}}} \frac{-1}{h_{i+\frac{1}{2}}} \frac{-1}{h_{i+\frac{1}{2}}} dx, \\ &= \frac{1}{h_{i-\frac{1}{2}}} + \frac{1}{h_{i+\frac{1}{2}}}, \quad \text{for } i \neq 1 \text{ or n}. \end{aligned} \tag{3.3.7}$$

Use the similar process, the i^th^ discrete equation for u'' is

$$-\left(\frac{1}{h_{i-\frac{1}{2}}} \right) u_{i-1} + \left(\frac{1}{h_{i-\frac{1}{2}}} + \frac{1}{h_{i+\frac{1}{2}}} \right) u_i - \left(\frac{1}{h_{i+\frac{1}{2}}} \right) u_{i+1}. \tag{3.3.8}$$

Consider another one dimensional problem with coefficients in the discrete operator

$$-(ku_x)_x = 0, \quad x \in \Omega. \tag{3.3.9}$$

If $k(x)$ has large jumps, interpolation used in GMG can yield poor performance for this problem. We discretize this equation with piecewise-linear finite elements. The mesh is shown in figure (3.11).

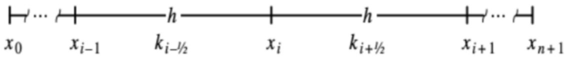

Figure 3.11: Discretization mesh for equation (3.3.9) *http://citeseerx.ist.psu.edu/viewdoc*

In this case, the i$^{\text{th}}$ discrete equation for $(ku_x)_x$ is

$$-\left(\frac{k_{i-\frac{1}{2}}}{h}\right)u_{i-1} + \left(\frac{k_{i-\frac{1}{2}}}{h} + \frac{k_{i+\frac{1}{2}}}{h}\right)u_i - \left(\frac{k_{i+\frac{1}{2}}}{h}\right)u_{i+1}. \tag{3.3.10}$$

with each k_m with no jumps. If the grid spacing satisfying $h_{i-\frac{1}{2}} = \frac{h}{k_{i-\frac{1}{2}}}$, and substitute $h_{i-\frac{1}{2}}$ and $h_{i+\frac{1}{2}}$ in (3.2.8) as $\frac{h}{k_{i-\frac{1}{2}}}$ and $\frac{h}{k_{i+\frac{1}{2}}}$ respectively, the second problem is equivalent to the first problem

$$\begin{aligned} -\left(\frac{k_{i-\frac{1}{2}}}{h}\right)u_{i-1} + \left(\frac{k_{i-\frac{1}{2}}}{h} + \frac{k_{i+\frac{1}{2}}}{h}\right)u_i - \left(\frac{h_{i+\frac{1}{2}}}{h}\right)u_{i+1} = 0, \\ \Rightarrow u_i = \left(\frac{k_{i+\frac{1}{2}}}{k_{i+\frac{1}{2}} + k_{i-\frac{1}{2}}}\right)u_{i-1} + \left(\frac{k_{i-\frac{1}{2}}}{k_{i+\frac{1}{2}} + k_{i-\frac{1}{2}}}\right)u_{i+1}. \end{aligned} \tag{3.3.11}$$

This example provides the key idea of AMG. For both of the examples, AMG only take into account the discretization matrix information, and construct the coarser levels based on these information. This resolves the problem when geometric information alone is not enough for some classes of problems, such as the second example. If considering the i$^{\text{th}}$ term in the discretization matrix of the two examples

$$(Au)_i = a_{i,i-1}u_{i-1} + a_{i,i}u_i + a_{i,i+1}u_{i+1}. \tag{3.3.12}$$

For the first example, $a_{i,i-1} = -\left(\frac{1}{h_{i-\frac{1}{2}}}\right)$, $a_{i,i} = \left(\frac{1}{h_{i-\frac{1}{2}}} + \frac{1}{h_{i+\frac{1}{2}}}\right)$, and $a_{i,i+1} = -\left(\frac{1}{h_{i+\frac{1}{2}}}\right)$. For the second example, $a_{i,i-1} = -\left(\frac{k_{i-\frac{1}{2}}}{h}\right)$, $a_{i,i} = \left(\frac{k_{i-\frac{1}{2}}}{h} + \frac{k_{i+\frac{1}{2}}}{h}\right)$, and $a_{i,i+1} = -\left(\frac{k_{i+\frac{1}{2}}}{h}\right)$.

3.3.2 AMG Algorithm

The symmetric positive definite (SPD) matrix is a symmetric matrix with all positive eigenvalues. Consider solving the linear system

$$A\mathbf{u} = \mathbf{f}. \tag{3.3.13}$$

where A is a SPD $n \times n$ matrix, and $\mathbf{u}$ and $\mathbf{f}$ are vectors in $\mathbb{R}^n$. AMG algorithm is originally developed based on the assumption that A is SPD. If A is not SPD, standard AMG will not work effectively. Let the fine-grid points with the indices $1, 2, ..., n$. According to Briggs et al. [1], the connections within grids are determined by the undirected adjacency graph of matrix A. For entry a_{ij} in A, if $a_{ij} \neq 0$, we view it as an association of vertex i and j in the graph. For example, in figure (3.12), Xs are non-zero entries in A, which are related to links in the undirected adjacency graph between nods. In this way, the grids and their connections can be entirely defined by matrix A.

$$A = \begin{bmatrix} X & X & & X & X & \\ X & X & X & X & & \\ & X & X & X & X & X \\ X & X & X & X & & \\ X & & X & & X & X \\ & & X & & X & X \end{bmatrix}$$

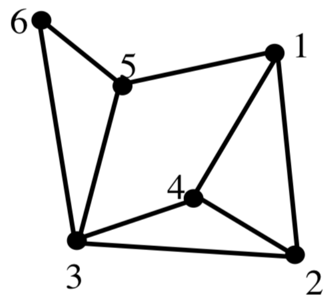

Figure 3.12: The matrix A and the related undirected adjacency graph. Xs are non-zero entries in A, and they are related to links in the undirected adjacency graph. *http://www.math.ust.hk/*

For selecting the coarse grid, AMG does not require the smooth functions to be geometrically smooth [1]. In GMG, we choose the coarse grid that represents the smooth functions accurately, and apply the intergrid operators that transfer the smooth functions between grids accurately. Instead, with AMG, we only select the coarse versions of operator A, and we do not have the physical grid.

The smoothness is defined algebraically, which is any error that is not reduced effectively by the relaxation process.

According to equation (3.1.7), we have

$$\mathbf{e}^m \leftarrow (I - Q^{-1}A)\mathbf{e}^{m-1}. \tag{3.3.14}$$

In this case, algebraic smoothness means the size of $\mathbf{e}^m$ is not significantly less than $\mathbf{e}^{m-1}$ [1]. If choosing

$$||\mathbf{e}||_A = (A\mathbf{e}, \mathbf{e})^{\frac{1}{2}}, \tag{3.3.15}$$

algebraically smooth errors should have the property that equation (3.3.14)

$$||(I - Q^{-1}A)\mathbf{e}^m||_A \approx ||\mathbf{e}^{m-1}||_A. \tag{3.3.16}$$

Let the restriction matrix be R and the prolongation matrix be P, the two-level AMG algorithm is

$$\mathbf{u}^h \leftarrow AMG(\mathbf{u}^h, \mathbf{f}^h)$$

- Perform k smoothing steps on $A^h\mathbf{u}^h = \mathbf{f}^h$ using classical iterative methods
- Compute residual $\mathbf{r}^h = \mathbf{f}^h - A^h\mathbf{u}^h = A^h\mathbf{e}^h$, and restrict it to the coarse grid by $\mathbf{r}^{2h} = R\mathbf{r}^h$
- Solve $A^{2h}\mathbf{e}^{2h} = \mathbf{r}^{2h}$ on Ω^{2h}
- Interpolate the coarse-grid error to the fine grid and correct $\mathbf{u}^h \leftarrow \mathbf{u}^h + Pe^{2h}$
- Do k smoothing steps on $A^h\mathbf{u}^h = \mathbf{f}^h$

In AMG, the grid levels are not corresponding to a real grid, which is different from GMG. To obtain the formula for the prolongation operator, consider dividing the points on the fine level into group C and group F. Points $i \in C$ are points on both the coarse-grid and the fine-grid, while points $i \in F$ are points on the fine-grid only. Suppose $e_i, i \in C$, represents the error that will be interpolated from the coarse-grid to the fine-grid. For $i \in F$, let the coarse-grid points that transfer information to i belong to C_i.

The prolongation operator P can be defined as

$$(Pe)_i = \begin{cases} e_i, \quad i \in C, \\ \\ \sum_{j\in C_i} \omega_{ij} e_j, \quad i \in F. \end{cases} \tag{3.3.17}$$

The error at the point on both coarse-grid and fine-grid is kept the same. For error at the point only on the fine-grid, we add the interpolation weights ω_{ij}.

Before the solution converges, there will be a vector associated with the approximated solution

$$A\mathbf{u}_{\text{approx}} - \mathbf{f} = -\mathbf{r}. \tag{3.3.18}$$

We seek a correction $\mathbf{u}_{\text{cor}}$ to $\mathbf{u}_{\text{approx}}$ so that the exact solution is given by

$$\mathbf{u} = \mathbf{u}_{\text{approx}} + \mathbf{u}_{\text{cor}}. \tag{3.3.19}$$

Substitute equation (3.3.19) to (3.3.13)

$$\begin{gathered} A(\mathbf{u}_{\text{approx}} + \mathbf{u}_{\text{cor}}) - \mathbf{f} = 0, \\ \Rightarrow A\mathbf{u}_{\text{cor}} + (\mathbf{u}_{\text{approx}} - \mathbf{f}) = 0. \end{gathered} \tag{3.3.20}$$

Substitute (3.3.18) to (3.3.20)

$$A\mathbf{u}_{\text{cor}} - \mathbf{r} = 0. \tag{3.3.21}$$

which is the equation for the correction in terms of the fine level operator A and the residual $\mathbf{r}$. To solve for the correction on the coarse level, we need to transfer the residual from the fine level to the coarse level using the restriction operator R.

$$A_{\text{c}}\mathbf{u}^{\text{c}}_{\text{cor}} - R\mathbf{r} = 0. \tag{3.3.22}$$

We use the solution to (3.3.22) to update the solution on the finer level

$$\mathbf{u}^{\text{new}}_{\text{approx}} = \mathbf{u}_{\text{approx}} + P\mathbf{u}^{\text{c}}_{cor} = 0. \tag{3.3.23}$$

where P is the prolongation operator. For AMG, the restriction and prolongation operator usually have the relationship

$$P = R^{\mathrm{T}}. \tag{3.3.24}$$

The coarse level operator A_{c} is constructed by the Galerkin approach. Since the residual associated with the corrected fine level solution should vanish when transferring back to the coarse level

$$R\mathbf{r}^{\mathrm{new}} = 0. \tag{3.3.25}$$

Substituting (3.3.13) and (3.3.23) for $\mathbf{r}^{\mathrm{new}}$ and $\mathbf{u}^{\mathrm{new}}_{\mathrm{approx}}$

$$\begin{aligned} R[A\mathbf{u}^{\mathrm{new}}_{\mathrm{approx}} - \mathbf{f}] &= 0, \\ \Rightarrow [A(\mathbf{u}_{\mathrm{approx}} + P\mathbf{u}^{\mathrm{c}}_{\mathrm{cor}}) - \mathbf{f}] &= 0. \end{aligned} \tag{3.3.26}$$

We can rearrange the second equation to

$$RAP\mathbf{u}^{\mathrm{c}}_{\mathrm{cor}} + R(A\mathbf{u}_{\mathrm{approx}} - \mathbf{f}]) = 0. \tag{3.3.27}$$

Thus

$$RAP\mathbf{u}^{\mathrm{c}}_{\mathrm{cor}} - R\mathbf{r} = 0. \tag{3.3.28}$$

Compare equation (3.3.28) with equation (3.3.22), the coarse level operator A_{c} is

$$A_{\mathrm{c}} = RAP = P^{\mathrm{T}}AP. \tag{3.3.29}$$

CHAPTER 4

Algorithmic Aspects of Multigrid Methods for the Bidomain Equations

4.1 Preconditioning

In numerical analysis, the condition number of a function measures the change of the output value when there is a small change in the input. For a linear system $A\mathbf{x} = \mathbf{b}$, if choosing a slightly different right hand side $\hat{\mathbf{b}}$, we will get a different solution $\hat{\mathbf{x}}$. The condition number of matrix A measures how much the relative error of the right hand side influence the relative error of the solution. We have $A(\hat{\mathbf{x}} - \mathbf{x}) = \hat{\mathbf{b}} - \mathbf{b}$, therefore

$$||\hat{\mathbf{x}} - \mathbf{x}|| = ||A^{-1}(\hat{\mathbf{b}} - \mathbf{b})|| \leq ||A^{-1}||||(\hat{\mathbf{b}} - \mathbf{b}||. \tag{4.1.1}$$

Since we have $||\mathbf{b}|| = ||A\mathbf{x}|| \leq ||A||||\mathbf{x}||$,

$$\frac{||\hat{\mathbf{x}} - \mathbf{x}||}{||\mathbf{x}||} \leq ||A||||A^{-1}||\frac{||(\hat{\mathbf{b}} - \mathbf{b})||}{||\mathbf{b}||}. \tag{4.1.2}$$

The number cond(A) $= ||A||||A^{-1}||$ is the condition number of matrix A. The idea of preconditioning for iterative solvers is to modify the A matrix in the linear system with another matrix $\hat{A}$ which has a smaller condition number. The standard formula for solving the left preconditioned system is to use a nonsingular matrix P

$$P^{-1}A\mathbf{x} = P^{-1}\mathbf{b}, \tag{4.1.3}$$

where $P^{-1}A$ is hopefully better conditioned than A. For the iterative methods

$$\begin{gathered} P^{-1}(A\mathbf{x} - \mathbf{b}) = P^{-1}\mathbf{r}, \\ \Rightarrow \mathbf{x}_{k+1}^{n} = \mathbf{x}_{k}^{n} + P^{-1}\mathbf{r}_{k}^{n}. \end{gathered} \tag{4.1.4}$$

The goal of using the preconditioned system is to reduce the condition number of the original system by considering the preconditioned matrix $P^{-1}A$ with a smaller condition number.

4.2 Block Preconditioners

To solve the bidomain system, we can use block factorization, in which case we split the discretization matrix into blocks. After that, we get two sub-systems $A\mathbf{x} = \mathbf{b}$, the two A matrix are the diagonal blocks of the original discretization of the bidomain system. In this chapter, I will implement block factorization for both GMG tests and AMG tests, and compare its performance with GMG and AMG direct solvers. We can write the bidomain system as

$$\begin{bmatrix} A_{VV} & A_{V\phi} \\ A_{\phi V} & A_{\phi\phi} \end{bmatrix} \begin{bmatrix} \mathbf{V}^n \\ \mathbf{\Phi}^n \end{bmatrix} = \begin{bmatrix} \mathbf{r}_{\mathrm{V}} \\ \mathbf{r}_{\phi} \end{bmatrix} \tag{4.2.1}$$

Without a bath, we have

$$\begin{aligned} A_{VV} &= \frac{\chi C_{\mathrm{m}}}{\triangle t} M + K_{\mathrm{i}}, \\ A_{V\phi} &= K_{\mathrm{i}}, \\ A_{\phi V} &= K_{\mathrm{i}}, \\ A_{\phi\phi} &= K_{\mathrm{i}} + K_{\mathrm{e}}. \end{aligned} \tag{4.2.2}$$

With a bath, the blocks are

$$\begin{aligned} A_{VV} &= \frac{\chi C_{\mathrm{m}}}{\triangle t} M + K_{i}, \\ A_{V\phi} &= \begin{bmatrix} K_i & 0 \end{bmatrix}, \\ A_{\phi V} &= \begin{bmatrix} K_{\mathrm{i}} \\ 0 \end{bmatrix}, \\ A_{\phi\phi} &= \begin{bmatrix} \kappa_{(1,1)} & \kappa_{(1,2)} \\ \kappa_{(2,1)} & \kappa_{(2,2)} \end{bmatrix}. \end{aligned} \tag{4.2.3}$$

We can define the residual as

$$\mathbf{r}_k^n = \begin{bmatrix} \mathbf{r}_{Vk}^n \\ \mathbf{r}_{\phi k}^n \end{bmatrix} = \begin{bmatrix} \mathbf{r}_V - A_{VV}\mathbf{V}_k^n - A_{V\phi}\mathbf{\Phi}_k^n \\ \mathbf{r}_\phi - A_{V\phi}\mathbf{V}_k^n - A_{\phi\phi}\mathbf{\Phi}_k^n \end{bmatrix}. \tag{4.2.4}$$

In the numerical experiments, we can use block Jacobi as the block preconditioner, in which case P is the diagonal of A

$$P = \begin{bmatrix} A_{VV} & 0 \\ 0 & A_{\phi\phi} \end{bmatrix}, \qquad P^{-1} = \begin{bmatrix} A^{-1}_{VV} & 0 \\ 0 & A^{-1}_{\phi\phi} \end{bmatrix}, \tag{4.2.5}$$

which means that

$$\begin{aligned} \mathbf{V}^n_{k+1} &= \mathbf{V}^n_k + A^{-1}_{VV}\mathbf{r}^n_{Vk} = \mathbf{V}^n_k + \hat{\mathbf{r}}^n_{Vk}, \\ \mathbf{\Phi}^n_{k+1} &= \mathbf{\Phi}^n_k + A^{-1}_{\phi\phi}\mathbf{r}^n_{\phi k} = \mathbf{\Phi}^n_k + \hat{\mathbf{r}}^n_{\phi k}. \end{aligned} \tag{4.2.6}$$

At every time-step, we solve the parabolic problem

$$A_{VV}\hat{\mathbf{r}}^n_{Vk} = \mathbf{r}^n_{Vk}. \tag{4.2.7}$$

and the elliptic problem

$$A_{\phi\phi}\hat{\mathbf{r}}^n_{\phi k} = \mathbf{r}^n_{\phi k}. \tag{4.2.8}$$

Another block preconditioner is the block Gauss-Seidel, in which case P is the lower triangular of the original A matrix

$$P = \begin{bmatrix} A_{VV} & 0 \\ A_{V\phi} & A_{\phi\phi} \end{bmatrix}, \qquad P^{-1} \begin{bmatrix} A^{-1}_{VV} & 0 \\ -A^{-1}_{\phi\phi}A_{V\phi}A^{-1}_{VV} & A^{-1}_{\phi\phi} \end{bmatrix}. \tag{4.2.9}$$

The block decomposition is

$$P^{-1} = \begin{bmatrix} I & 0 \\ 0 & A^{-1}_{\phi\phi} \end{bmatrix}, \begin{bmatrix} I & 0 \\ -A_{\phi V} & I \end{bmatrix} \begin{bmatrix} A^{-1}_{\mathrm{VV}} & 0 \\ 0 & I \end{bmatrix}. \tag{4.2.10}$$

In this way

$$\begin{aligned} \mathbf{V}^n_{k+1} &= \mathbf{V}^n_k + A^{-1}_{VV}\mathbf{r}^n_{Vk} = \mathbf{V}^n_k + \hat{\mathbf{r}}^n_{Vk}, \\ \mathbf{\Phi}^n_{k+1} = \mathbf{\Phi}^n_k - A^{-1}_{\phi\phi}A_{V\phi}A^{-1}_{VV}\mathbf{r}^n_{Vk} + A^{-1}_{\phi\phi}\mathbf{r}^n_{\phi k} &= \mathbf{\Phi}^n_k + A^{-1}_{\phi\phi}(-A_{\phi V}\hat{\mathbf{r}}^n_{Vk} + \mathbf{r}^n_{\phi k}) = \mathbf{\Phi}^n_k + \hat{\mathbf{r}}^n_{\mathrm{new}\ k}. \end{aligned} \tag{4.2.11}$$

At each step we solve the parabolic problem

$$A_{VV}\hat{\mathbf{r}}^n_{Vk} = \mathbf{r}^n_{Vk}. \tag{4.2.12}$$

and the elliptic problem

$$A_{\phi\phi}\hat{\mathbf{r}}^n_{\text{new }k} = -A_{\phi V}\hat{\mathbf{r}}^n_{Vk} + \mathbf{r}^n_{\phi k}. \tag{4.2.13}$$

4.3 Simulation

4.3.1 GMG Performance for Simple 2D Geometries

This section considers the finite-difference discretization for the bidomain system. I will apply GMG as a solver, GMG as a preconditioner for GMRES for the whole system, as well as GMG as a preconditioner for the block factorization to solve the bidomain equations. Consider the discretized bidomain system, to apply GMG, I perform k smoothing steps with classical iterative methods, such as SOR. Decompose matrix A into the diagonal component D, the strictly lower triangular matrix L, and the strictly upper triangular matrix U,

$$A = D + L + U. \tag{4.3.1}$$

We have

$$\begin{aligned}
A\mathbf{u} - \mathbf{f} &= 0, \\
\Rightarrow \omega(A\mathbf{u} - \mathbf{f}) - 0, \\
\Rightarrow D\mathbf{u} = D\mathbf{u} + \omega(A\mathbf{u} - \mathbf{f}), \\
\Rightarrow D\mathbf{u} = D\mathbf{u} + \omega(L + U - D)\mathbf{u} - \omega\mathbf{f}, \\
\Rightarrow (D - \omega L)\mathbf{u} = (+(1-\omega)D)\mathbf{u} + \omega(L + U - D)\mathbf{u} - \omega\mathbf{f}, \\
\Rightarrow (D - \omega L)\mathbf{u}^n = (+(1-\omega)D)\mathbf{u} + \omega(L + U - D)\mathbf{u}^{n-1} - \omega\mathbf{f}.
\end{aligned} \tag{4.3.2}$$

The iteration matrix is

$$P_{\text{SOR}} = (D - \omega L)^{-1}(\omega U + (1-\omega)D). \tag{4.3.3}$$

If $\omega = 1$, the SOR simplifies to the Gauss-Seidel method, which is

$$\begin{aligned}
A\mathbf{u} - \mathbf{f} &= 0, \\
(D - L)\mathbf{u}^n &= U\mathbf{u}^{n-1} + \mathbf{f}.
\end{aligned} \tag{4.3.4}$$

which I will use as the smoother for GMG in the numerical simulation.

In the numerical tests of the anisotropy bidomain system without bath, I use $C_\text{m} = 1\ \mu\text{Fcm}^{-2}, \chi =$

1400 cm^{-1}, and

$$\sigma_{\mathrm{i}} = \begin{bmatrix} 1.7 & 0 \\ 0 & 0.19 \end{bmatrix}, \qquad \sigma_{\mathrm{e}} = \begin{bmatrix} 6.2 & 0 \\ 0 & 2.4 \end{bmatrix}, \qquad \sigma_{\mathrm{b}} = 20, \tag{4.3.5}$$

the units is mScm^{-1}. Denote this set of conductivity values as σ_1, which is from Clerc's work [66]. In the bidomain system, I_{ion}, is the sum of ionic current I of specific types of ions. In the numerical simulation, I use

$$I_{\mathrm{ion}} = (V - V_{\mathrm{depolarization}})(V - V_{\mathrm{rest}})(V - V_{\mathrm{threshold}}), \tag{4.3.6}$$

where $V_{\mathrm{depolarization}}$ is maximum transmembrane voltage at depolarization, V_{rest} is the transmembrane voltage at the resting condition, and $V_{\mathrm{threshold}}$ is the transmembrane voltage when the sodium channel opens. In the numerical experiment, $V_{\mathrm{depolarization}} = 30$ mV, $V_{\mathrm{rest}} = -85$ mV, and $V_{\mathrm{threshold}} = -57$ mV. The parameters are similar as those in Pathmanathan et al. [20]. The domain size is 2 cm$\times$2 cm, and the mesh is discretized uniformly. Within a square of 1 cm$\times$1 cm at the center of the mesh, the initial value of V is set to 30 mV, which is similar as the maximum transmembrane voltage difference during depolarization. Outside that square, the initial value of V is set to -85 mV, which is the transmembrane voltage at the resting condition (figure (4.1)). At each GMG levels, the operators are directly re-discretized using FDM. As the grids in x and y directions of the coarser level are half of those of the finer level, if the discretization matrix of an operator in the finer level is of size $N \times N$, that of the coarser level is of size $\frac{N}{2} \times \frac{N}{2}$.

Figure (4.2) shows the relative residual plot for 32×32, 64×64, 128×128, 256×256, and 512×512 grids with with Neumann boundary conditions, for both the GMG solver and GMG as a preconditioner for GMRES. The relative tolerance is set as 10^{-12} for the GMG preconditioner, and the number of V-cycles for the GMG solver is 20. The number of sweeps at each GMG level is 2. For the GMG preconditioner, I only use 1 V-cycle at each GMRES iteration. The convergence iterations for different time-steps are shown in table (4.1).

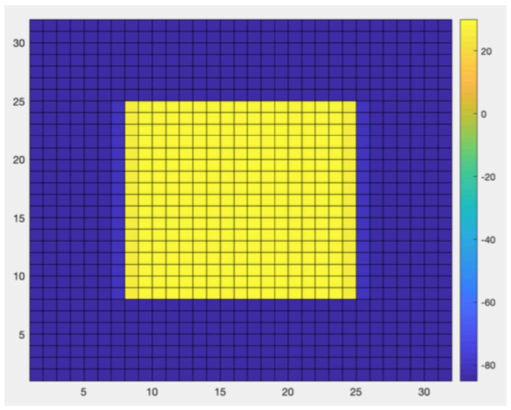

Figure 4.1: Initial condition for the bidomain system without bath using a 32×32 grid

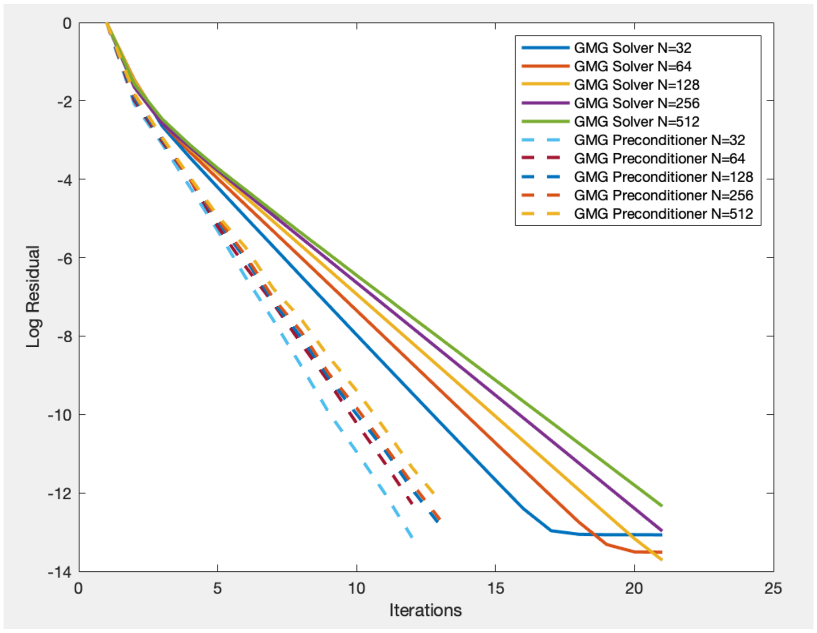

Figure 4.2: Relative residual plot for GMG solver and GMG preconditioner for GMRES with σ_1 for the bidomain without bath system, with $\Delta t = 0.0125$ ms

Grid	$\Delta t = 0.05$ ms	$\Delta t = 0.0125$ ms
32×32	13	12
64×64	14	12
128×128	14	13
256×256	15	13
512×512	15	13

Table 4.1: Convergence iterations for GMG as a preconditioner approach with σ_1 for the bidomain without bath system

The solution plots V and ϕ_e at $t = 1$ ms, and $t = 5$ ms are shown in figure (4.3).

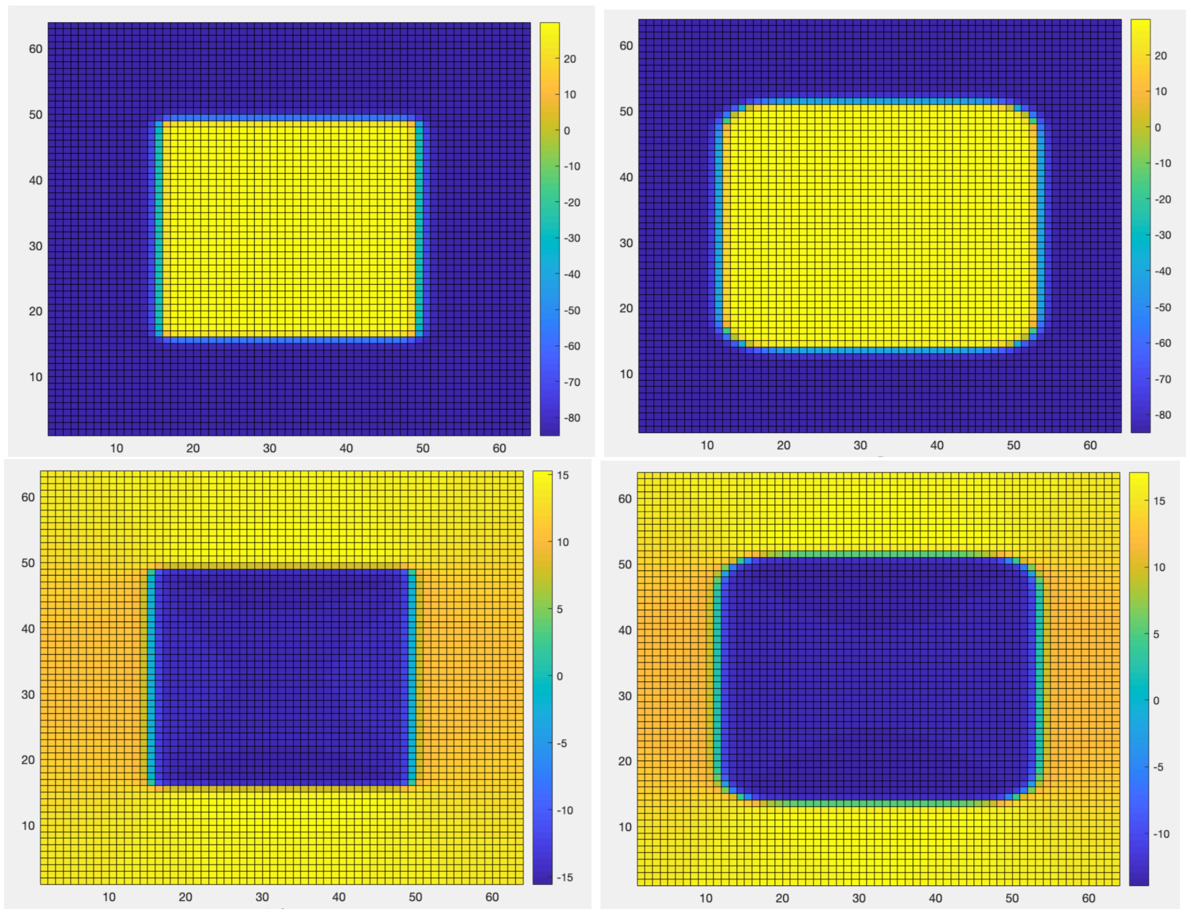

Figure 4.3: Solution plots at $t = 1$ ms, and $t = 5$ ms (from left to right) showing V (top) and ϕ_e (bottom)

According to the figures, the propagation along the x direction is faster than along the y

direction. This is because the σ in the x direction is greater than that in the y direction.

I also tried a different set of conductivity tensor values, as suggested in Granham et al. [67].

$$\sigma_{\mathrm{i}} = \begin{bmatrix} 2.8. & 0 \\ 0 & 0.26 \end{bmatrix}, \qquad \sigma_{\mathrm{e}} = \begin{bmatrix} 2.2 & 0 \\ 0 & 1.3 \end{bmatrix}, \tag{4.3.7}$$

in units mScm^{-1}. Denote this set of conductivity values as σ_2. Figure (4.4) and table (4.2) show the relative residual plot, as well as the convergence iterations.

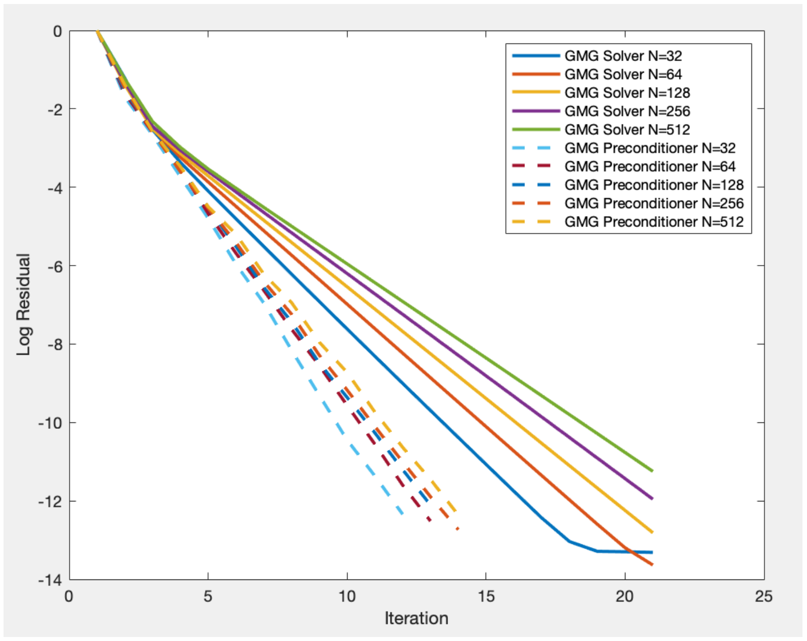

Figure 4.4: Relative residual plot for GMG solver and GMG preconditioner for GMRES with σ_2 for the bidomain without bath system, with $\Delta t = 0.0125$ ms

Grid	$\Delta t = 0.05$ ms	$\Delta t = 0.0125$ ms
32×32	13	12
64×64	15	13
128×128	16	13
256×256	18	14
512×512	19	14

Table 4.2: Convergence iterations for GMG as a preconditioner approach with σ_2 for the bidomain without bath system

According to the results, the convergence iterations for $\Delta t = 0.05$ ms are greater than those for $\Delta t = 0.0125$ ms. For different anisotropy ratios, the convergence iterations do not differ significantly for $\Delta t = 0.0125$ ms. Using GMG as a preconditioner for GMRES converges more rapidly than using GMG as a solver. For the same grids with the same time-step size, the convergence iterations of GMG preconditioner are 4-5 less than those of the the GMG solver.

For the bidomain with bath problem, a bath of size 2 cm$\times$2 cm is added to the right of the muscle part (figure(4.1)). The initial condition in the bath region is 0 mV. The grid spacing in the muscle part is the same as in the bath part. The numerical results with σ_1 [66] are shown in figure (4.5) and table (4.3).

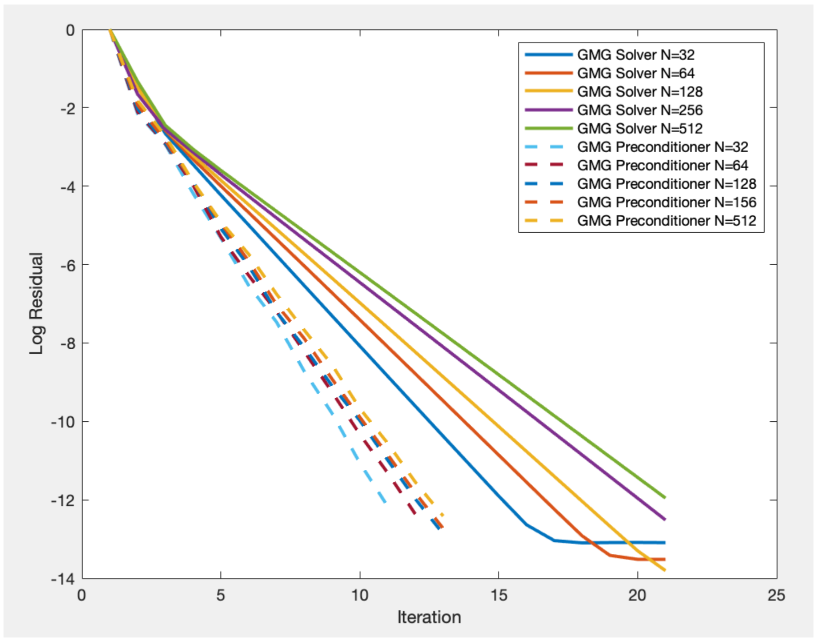

Figure 4.5: Relative residual plot for GMG solver and GMG preconditioner for GMRES with σ_1 for the bidomain with bath system, with $\Delta t = 0.0125$ ms

Grid	$\Delta t = 0.05$ ms	$\Delta t = 0.0125$ ms
32×32	12	11
64×64	13	12
128×128	13	13
256×256	14	13
512×512	14	13

Table 4.3: Convergence iterations for GMG as a preconditioner approach with σ_1 for the bidomain with bath system

According to the results, the convergence rates for the with bath case are similar as the without bath case with σ_1. The iterations do not increase significantly as the number of grids increases.

In addition to GMG as a direct solver and as a preconditioner for GMRES, I also experiment with the block factorization approach. I use the block Jacobi as the preconditioner for GMRES. For each block, I use GMG as the preconditioner, in which case I perform 1 V-cycle of GMG for each

GMRES iteration. The results for the without and with bath cases are shown in figure (4.6) and (4.7) and table (4.4) and (4.5).

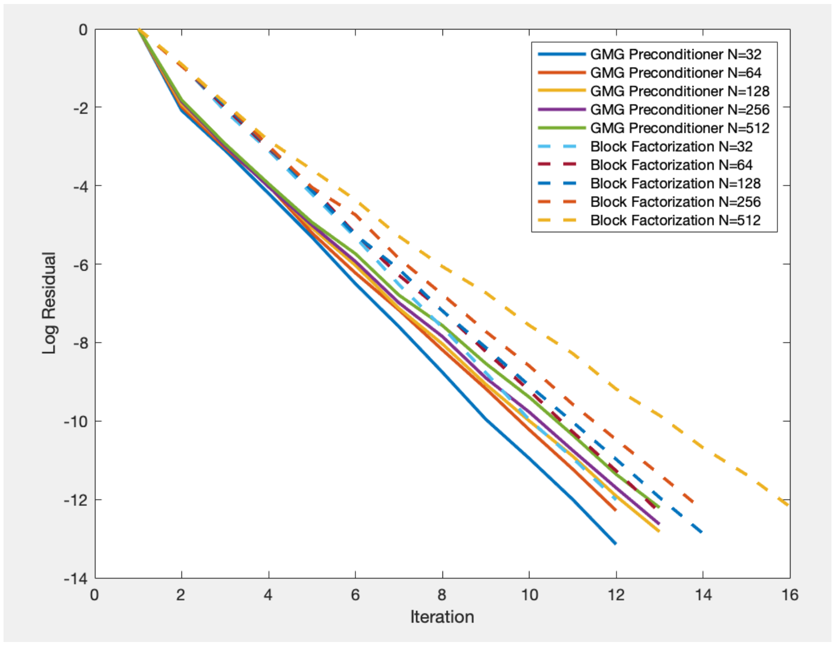

Figure 4.6: Relative residual plot for GMG solver and GMG preconditioner for GMRES with σ_1 for the bidomain without bath system, with $\Delta t = 0.0125$ ms

Grid	$\Delta t = 0.05$ ms	$\Delta t = 0.0125$ ms
32×32	12	12
64×64	13	13
128×128	14	14
256×256	14	14
512×512	16	16

Table 4.4: Convergence iterations for block factorization approach with σ_1 for the bidomain without bath system

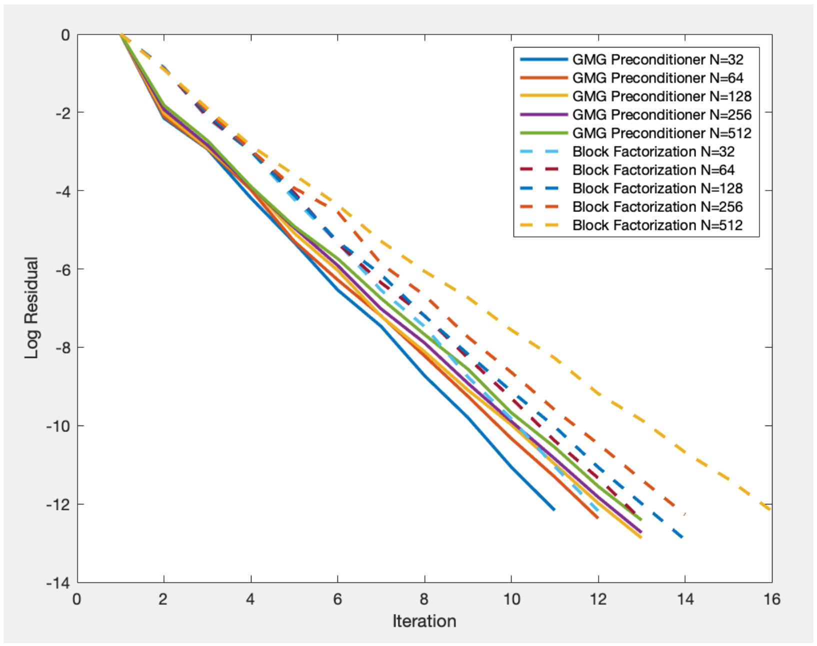

Figure 4.7: Relative residual plot for block factorization approach with σ_1 for the bidomain with bath system, with $\Delta t = 0.0125$ ms

Grid	$\Delta t = 0.05$ ms	$\Delta t = 0.0125$ ms
32×32	13	12
64×64	13	13
128×128	14	14
256×256	15	14
512×512	16	16

Table 4.5: Convergence iterations for block factorization approach with σ_1 for the bidomain with bath system

According to the results, the convergence iterations increase by 1 or 2 for both the with and without bath cases as the number of grids increases. The number of iterations is smaller than that of the GMG solver, while slightly greater than that of the GMG preconditioner.

I also considered the fibrosis case. Fibrosis is the situation of the expansion of extracellular matrix and the increasing of the number of fibroblasts [68]. The existance of fibrosis forces electrical propagation to take a zigzag pattern, which slows the conduction velocity [69] (figure 4.8). In numerical experiments, I generated random numbers between 0 and 1, and related different numbers to the entries of the discretization matrix. I studied 30%, 70%, and 90% of nodes related to fibrosis. I only consider the fibrosis in the intra-cellular space. If the random number is less than that percentage, the associated matrix entry becomes zero. For each MG level, I re-discretize the operators.

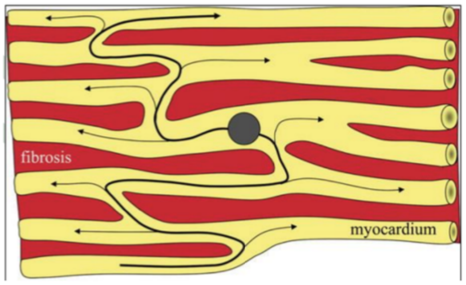

Figure 4.8: The existence of fibrosis (red region) forces electrical propagation to take a zigzag pattern, which slows the conduction velocity *https://www.semanticscholar.org/paper/Fibrosis-and-cardiac-arrhythmias.-Jong-Veen/06bfb1ad1ac8028a6baa5277b215e5128f81a2a1/figure/2*

Table (4.6) show the results for bidomain without bath problem, and table (4.7) shows the with bath case. The results are using the GMG preconditioner for GMRES.

	30%	70%	90%
32×32	12	11	11
64×64	12	12	11
128×128	13	12	11
128×128	13	12	12
256×256	13	12	12
512×512	14	13	12

Table 4.6: Convergence iterations for the fibrosis case with GMG as a preconditioner approach with σ_1 and $\Delta t = 0.0125$ ms for the bidomain without bath system

	30%	70%	90%
32×32	12	12	11
64×64	12	12	11
128×128	12	12	11
256×256	13	12	12
512×512	14	13	12

Table 4.7: Convergence iterations for the fibrosis case with GMG as a preconditioner approach with σ_1 and $\Delta t = 0.0125$ ms for the bidomain with bath system

According to the tables, the results for the with and without bath cases do not differ substantially when considering the fibrosis. The convergence rate for the 90% fibrosis is slightly higher compared with the other cases considered here.

Figure (4.8) shows the 70% fibrosis solution plot.

`fibrosisplot.pdf`

Figure 4.9: Solution plot for 70% at $t = 1$ ms fibrosis

I also explore using the block factorization approach for the fibrosis problem. Table (4.8) and (4.9) show the results with and without baths.

	30%	70%	90%
32×32	12	12	11
64×64	13	12	12
128×128	13	13	12
256×256	14	14	12
512×512	15	14	13

Table 4.8: Convergence iterations for the fibrosis case with the block factorization approach with σ_1 and $\Delta t = 0.0125$ ms for the bidomain without bath system

	30%	70%	90%
32×32	12	12	11
64×64	13	12	12
128×128	13	13	13
256×256	14	13	13
512×512	16	14	13

Table 4.9: Convergence iterations for the fibrosis case with the block factorization approach with σ_1 and $\Delta t = 0.0125$ ms for the bidomain with bath system

The results for block factorization shows the similar trend as using the GMG preconditioner for GMRES. The convergence iterations do not differ significantly of different fibrosis percentage. The convergence rate for the 90% fibrosis is the highest. Convergence iterations of GMG preconditioner is about 0-2 less than that of the block factorization with the same grids and the same fibrosis percentage, for the without and with bath cases.

4.3.2 AMG Performance for Simple 2D Geometries

In this section, I will present the results for applying AMG to the bidomain equations with and without bath. The AMG algorithm is provided by PETSc (GAMG). I will show results of both the block factorization approach and AMG applied to the full bidomain system.

As in the previous section, I set the domain size as 2 cm×2 cm. The initial condition is set so that if 0.5 cm $< x, y <$ 1.5 cm, $V = 30$ mV, otherwise $V = 85$ mV. The 1 cm×1 cm square is the initially activated region(figure (4.10)). I performed numerical experiments with $16 \times 16, 32 \times 32, 64 \times 64, 128 \times 128, 256 \times 256, 512 \times 512, 1024 \times 1024$ grids, and with $t = 0.5, 0.0125$ ms. For the block factorization approach, I considered both block Jacobi and block Gauss-Seidel to split the bidomain system into blocks. For each block, I used SOR or AMG. For the AMG applied to the full bidomain system, I used SOR as the smoother. The relative tolerance is 10^{-12}.

In the numerical simulations, I used the finite element discretization. I tested both the four-noded rectangular element (QUAD4) and the three-noded triangular element (TRI3). Table (4.10)(4.11) and figure (4.11)(4.13) show the results with conductivity σ_1. In the tables, BGS_S/G denotes the block Gauss-Seidel with SOR preconditioning the parabolic block and AMG preconditioning for the elliptic block. J_S/G denotes the block Jacobi with SOR preconditioning the parabolic block and AMG preconditioning for the elliptic block. BGS_G/G denotes the block Gauss-Seidel with AMG preconditioning for both of the parabolic and the elliptic block. AMG denotes AMG applied to the full bidomain system.

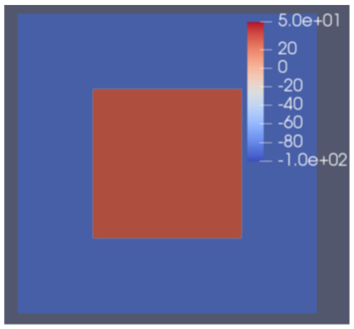

Figure 4.10: Initial condition plot for the bidomain without bath tests

$\Delta t = 0.05$ms	BGS_S/G	J_S/G	BGS_G/G	AMG
16×16	7	8	8	7
32×32	8	8	9	12
64×64	11	11	12	18
128×128	13	13	13	35
256×256	15	15	15	86
512×512	16	16	16	163
1024×1024	18	18	18	231
$\Delta t = 0.0125$ms	BGS_S/G	J_S/G	BGS_G/G	AMG
16×16	7	7	7	7
32×32	8	8	8	12
64×64	10	10	11	25
128×128	12	12	12	16
256×256	13	13	13	56
512×512	14	14	15	155
1024×1024	15	15	16	286

Table 4.10: Convergence iterations with σ_1 for the bidomain without bath system, with element type QUAD4

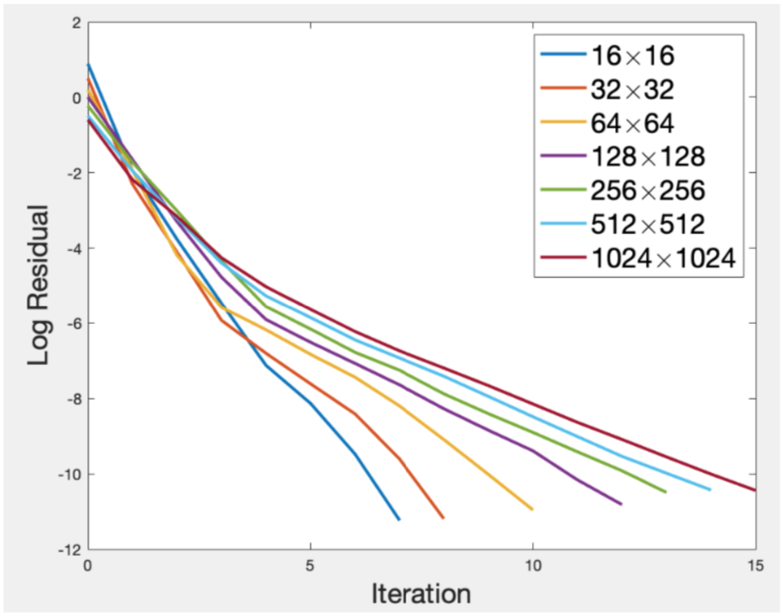

Figure 4.11: Iteration plot for BGS_G/G and σ_1 for the bidomain without bath system, with $\Delta t = 0.0125$ ms and element type QUAD4

Solution plots at $t = 1$ ms of V and ϕ_e are shown in figure (4.12). The ionic current propagation are similar as in figure (4.3).

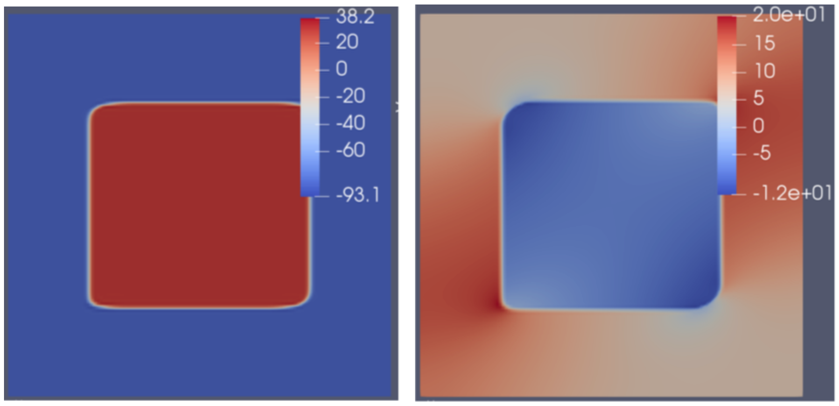

Figure 4.12: Solution plots at $t = 1$ ms of V (left) and ϕ_e (right)

$\Delta t = 0.05$ms	BGS_S/G	J_S/G	BGS_G/G	AMG
16×16	8	6	8	9
32×32	9	8	9	14
64×64	9	10	10	21
128×128	11	11	11	51
256×256	12	12	12	120
512×512	14	14	14	160
1024×1024	16	16	16	288
$\Delta t = 0.0125$ms	BGS_S/G	J_S/G	BGS_G/G	AMG
16×16	6	6	7	9
32×32	7	7	8	15
64×64	8	8	9	25
128×128	9	9	10	38
256×256	11	11	11	84
512×512	12	12	12	191
1024×1024	13	13	13	267

Table 4.11: Convergence iterations with σ_1 for the bidomain without bath system, with element type TRI3

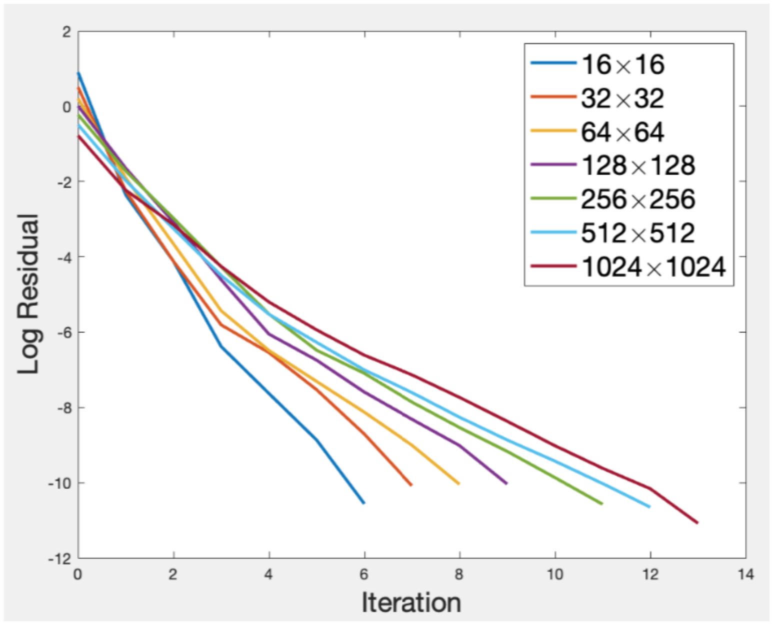

Figure 4.13: Iteration plot for BGS_G/G and σ_1 for the bidomain without bath system, with $\Delta t = 0.0125$ ms and element type TRI3

According to the results, the number of iterations required to reach 10^{-12} increase slowly with the increase of the number of grids with the block factorization approach. The convergence iterations are similar for the block algorithms. The AMG solver works poorly especially for the larger grid spacing. Thus I only tested block Gauss-Seidel with AMG preconditioning for both the parabolic and the elliptic block for the other tests in this section.

Figure (4.14) and table (4.12) show the results with σ_2 [67] for the two types of elements.

QUAD4	$\Delta t = 0.05$ ms	$\Delta t = 0.0125$ ms	TRI3	$\Delta t = 0.05$ ms	$\Delta t = 0.0125$ ms
16×16	7	7		7	6
32×32	9	8		9	8
64×64	12	10		13	10
128×128	14	13		14	11
256×256	16	15		15	13
512×512	16	16		15	14
1024×1024	18	18		17	16

Table 4.12: Convergence iterations for BGS_G/G with σ_2 for the bidomain without bath system

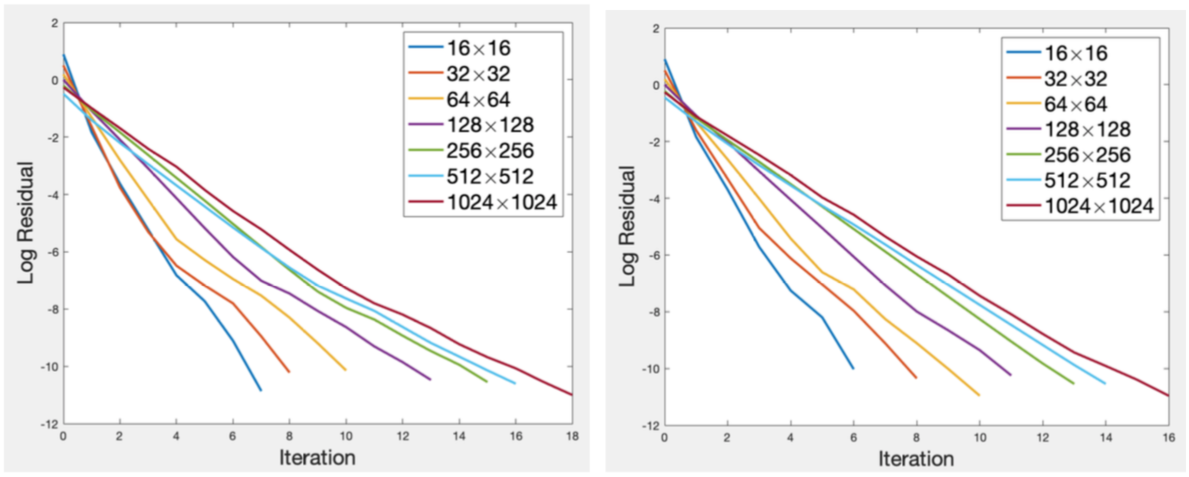

Figure 4.14: Iteration plot for BGS_G/G and σ_2 for the bidomain without bath system, with $\Delta t = 0.0125$ ms, for QUAD4 (left) and TRI3 (right)

From the results, the convergence iterations with $\Delta t = 0.05$ ms are slightly more than those with $\Delta t = 0.0125$ ms, which is similar as in section 4.2. In addition, the convergence iterations with σ_2 are not very different from with σ_1, indicating that the block factorization approach is efficient for different conductivities.

For the bidomain with bath problem, I divided the domain into two equal-spaced regions, representing the bath and the muscle. In the muscle part, the initially activated region is of size 0.5 cm×1 cm (figure (4.15)).

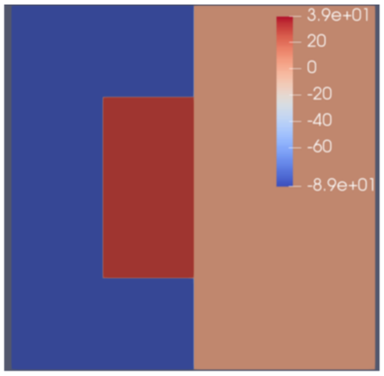

Figure 4.15: Initial condition plot for the bidomain with bath tests

Table (4.13) and figure (4.16) show the results with σ_1 and the plots for $\Delta t = 0.0125$ ms.

QUAD4	$\Delta t = 0.05$ ms	$\Delta t = 0.0125$ ms	TRI3	$\Delta t = 0.05$ ms	$\Delta t = 0.0125$ ms
16×16	6	6		6	6
32×32	7	7		7	7
64×64	10	8		8	7
128×128	11	10		9	9
256×256	12	11		10	9
512×512	13	12		11	10
1024×1024	16	14		13	11

Table 4.13: Convergence iterations for BGS_G/G with σ_1 for the bidomain with bath system

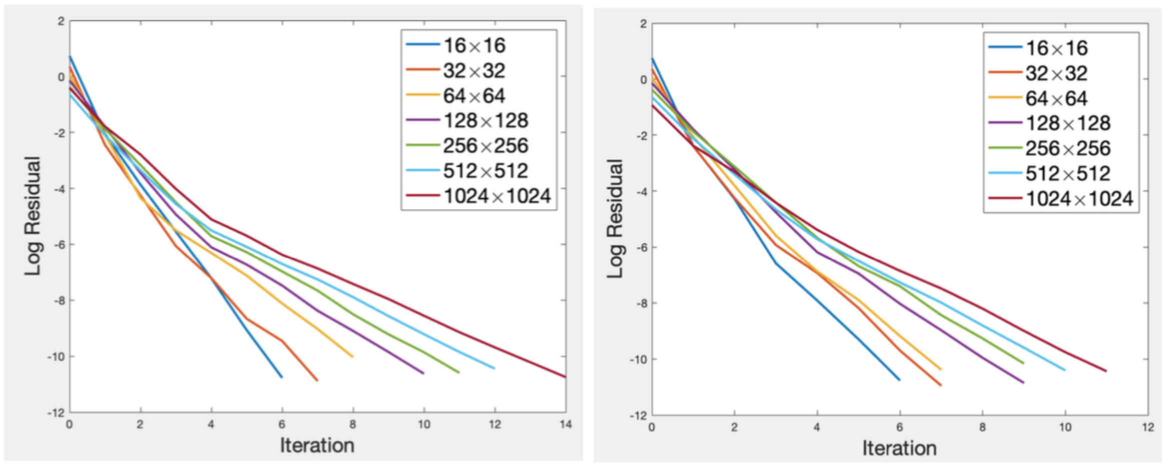

Figure 4.16: Iteration plot for BGS_G/G and σ_1 for the bidomain with bath system, with $\Delta t = 0.0125$ ms, for QUAD4 (left) and TRI3 (right)

Figure (4.17) shows the solution plot of V, ϕ_e, and ϕ_b at $t = 1$ ms. The ionic current propagation matches that in section 4.3.1.

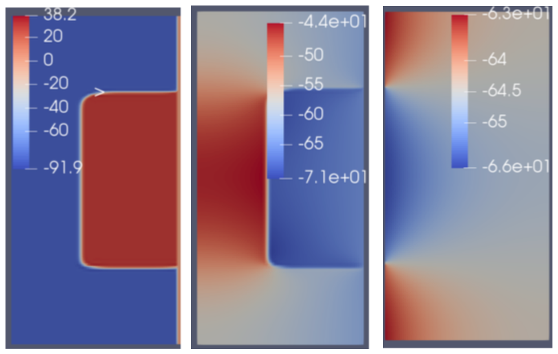

Figure 4.17: Solution plots at $t = 1$ ms of V (left), ϕ_e (middle), and ϕ_b (right)

Table (4.14) and figure (4.18) show the results with σ_2 [67] and the plots for $\Delta t = 0.0125$ ms.

QUAD4	$\Delta t = 0.05$ ms	$\Delta t = 0.0125$ ms	TRI3	$\Delta t = 0.05$ ms	$\Delta t = 0.0125$ ms
16×16	7	6		7	6
32×32	8	7		8	7
64×64	11	8		11	8
128×128	13	10		13	10
256×256	14	13		14	13
512×512	14	14		14	14
1024×1024	17	16		16	15

Table 4.14: Convergence iterations for BGS_G/G with σ_2 for the bidomain with bath system

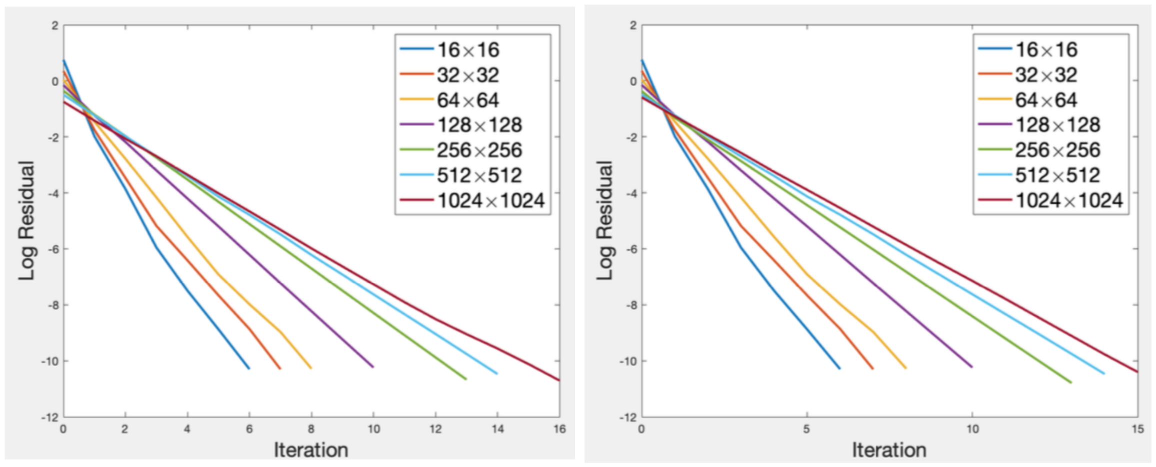

Figure 4.18: Iteration plot for BGS_G/G and σ_2 for the bidomain with bath system, with $\Delta t = 0.0125$ ms, for QUAD4 (left) and TRI3 (right)

According to the tables, the convergence iterations for the different elements do not differ significantly, which is similar as the without bath case. The convergence iterations of the with and without bath cases are also close to each other, suggesting that block factorization works efficiently for the two cases.

Besides the grid aligned case, I also tested the non-aligned anisotropy. In this situation, the anisotropic direction is not aligned with the grid discretization. The direction I used was with $\theta = 22.5°$. The results for the without bath (table (4.15) and figure (4.19)) and without bath (table (4.16) and figure (4.20)) are shown. The solution plots for the with base case at $t = 1$ ms, $t = 5$ ms, and $t = 10$ ms are shown in figure (4.21).

	$\Delta t = 0.05$ ms	$\Delta t = 0.0125$ ms
16×16	8	7
32×32	9	8
64×64	11	9
128×128	13	11
256×256	14	13
512×512	15	14
1024×1024	15	15

Table 4.15: Convergence of the non-aligned case with BGS_G/G and σ_1 for the bidomain without bath system, with QUAD4

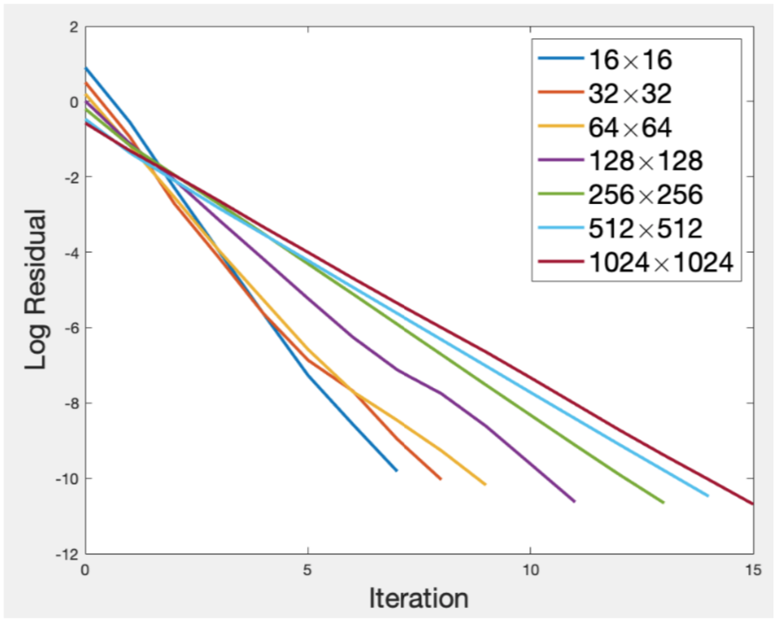

Figure 4.19: Iteration plot or the non-aligned case with BGS_G/G and σ_1 for the bidomain without bath system, with $\Delta t = 0.0125$ ms and QUAD4

	$\Delta t = 0.05$ ms	$\Delta t = 0.0125$ ms
16×16	7	7
32×32	9	8
64×64	10	9
128×128	12	10
256×256	14	13
512×512	15	14
1024×1024	15	14

Table 4.16: Convergence the non-aligned case with BGS_G/G and σ_2 for the bidomain with bath system, with QUAD4

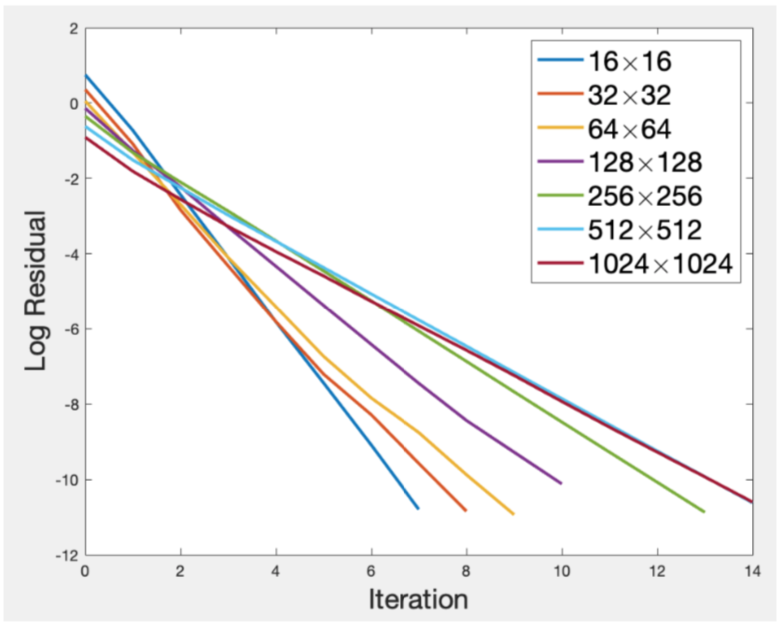

Figure 4.20: Iteration plot or the non-aligned case with BGS_G/G and σ_2 for the bidomain with bath system, with $\Delta t = 0.0125$ ms and QUAD4

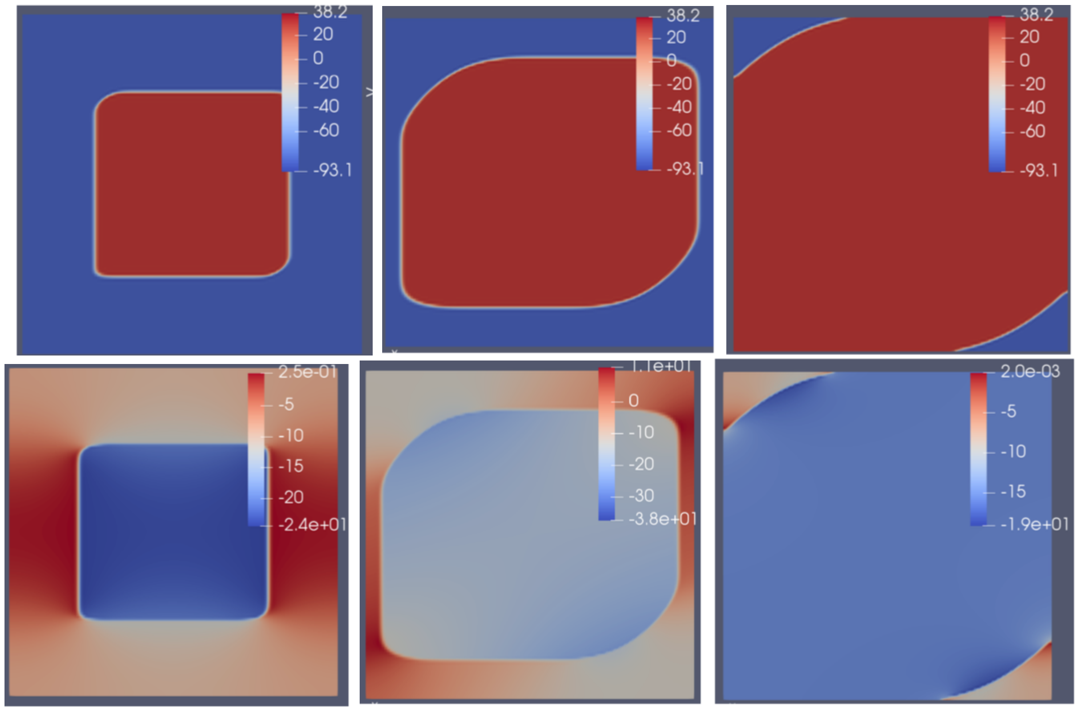

Figure 4.21: Solution plots at $t = 1, 5, 10$ ms (from left to right) of V (top) and ϕ_e (bottom), for the non-aligned without bath case

In the solution plot, the ionic current propagates quicker in the lower-left-corner than to the upper-right-corner direction, which is due to the tensor value with 22.5° of the direction of the grid discretization. According to the results, the non-aligned case yields similar convergence iterations with both σ_1 and σ_2. The convergence iterations are slightly less for the non-aligned situations than for the aligned case.

The results for the fibrosis cases without bath are shown in table (4.17).

$\Delta t = 0.0125$ms	30%	70%	90%
16×16	6	6	6
32×32	7	7	7
64×64	8	8	8
128×128	9	9	8
256×256	11	10	10
512×512	11	12	11
1024×1024	14	13	12

Table 4.17: Convergence iterations for $30\%, 70\%, 90\%$ fibrosis with BGS_G/G, $\Delta t = 0.0125$ ms, σ_1, and QUAD4, for the without bath case

The results for the fibrosis cases with bath are shown in table (4.18).

$\Delta t = 0.0125$ms	30%	70%	90%
16×16	6	6	6
32×32	7	7	7
64×64	8	7	7
128×128	9	9	8
256×256	10	10	9
512×512	11	11	10
1024×1024	13	13	12

Table 4.18: Convergence iterations for $30\%, 70\%, 90\%$ fibrosis with BGS_G/G, $\Delta t = 0.0125$ ms, σ_1, and QUAD4, for the with bath case

According to the results, the convergence iterations of 30% and 70% fibrosis cases are quite similar, while those of 90% fibrosis are less than the other cases.

Solution plots at $t = 1$ ms of V for the 70% fibrosis are shown in figure (4.22).

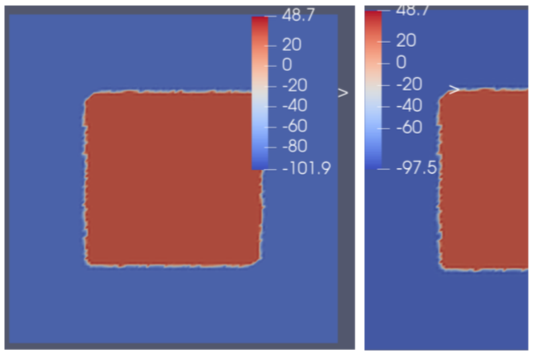

Figure 4.22: Solution plots of V for the the 70% fibrosis case at $t = 1$ ms of V of the without bath case and the with bath case

From the solution plot, we can view the fuzzy boundary of the activated region, which is due to the fibrosis in the simulation. According to the result tables, the convergence iterations of the with and without bath cases are similar. The convergence iterations of 90% fibrosis case are about 1 iteration smaller than the other fibrosis cases. Also, the convergence iterations of all the fibrosis cases are 1-3 iterations less than the without fibrosis tests.

4.4 Conclusion

In this section, I compared the performance of GMG solver, GMG preconditioner for GMRES, and block factorization with GMG preconditioner for both blocks on the idealized 2D geometry, both with and without bath. For the non-fibrosis case, the GMG solver yields the highest convergence iterations, while the GMG preconditioner yields the lowest. Convergence iterations only grow about 0-2 as the grid doubles in x and y directions for the GMG preconditioner and the block factorization approach. The convergence iterations of the block factorization is 1-3 more than those of the GMG preconditioner, and is 2-4 fewer than those of the the GMG solver, with the same time-step size and grids. For the fibrosis cases, the 90% fibrosis case yields the least convergence iterations.

In the AMG tests, the AMG solver performs poorly, while the block factorization schemes yield few convergence iterations. For both the with and without bath case, the convergence iterations with $\Delta t = 0.0125$ ms increase 0-2 as the grid doubles in x and y directions. In addition, similar as in the GMG tests, the convergence of the 90% fibrosis case are about 1 iteration smaller than the other cases with the same time-step size and grids. Also, the convergence iterations of all the fibrosis cases are fewer than those of the without fibrosis cases.

In the tests in this section, block factorization converges significantly faster than the AMG/GMG solvers. One reason is that AMG/GMG work best for Poisson-like problems. After splitting the system into blocks, each sub-block resembles a Poisson-like form, which is ideal for the AMG/gMG preconditioner.

CHAPTER 5
MG Performance for Realistic Three-Dimensional Geometries

5.1 Simulation Results for Real Geometry

In this section, I will show results obtained using a realistic three-dimensional geometry for the bidomain model with bath. The mesh I use represents the whole posterior wall of the left atrium. The detailed geometry was collected by fast anatomical mapping with a $2-5-2$ PentaRay catheter [70]. The muscle region is 1.5 mm, and the bath region is 2.85 mm. I will use the AMG for the entire system as well as the block factorization approach. The meshes I use are shown in figure (5.1), starting from the left corner, they are named are mesh0 (the coarsest), mesh 1, mesh 2, mesh 3, and mesh 4 (the finest). The meshes are unstructured, with different elements in the muscle (blue) and bath (red). Except for mesh 0 and mesh 1, the muscle part is finer than the bath part on the meshes. This is due to the purpose of considering the real heart, which has relative large bath region compared with muscle. A coarse mesh in the bath region saves computational cost.

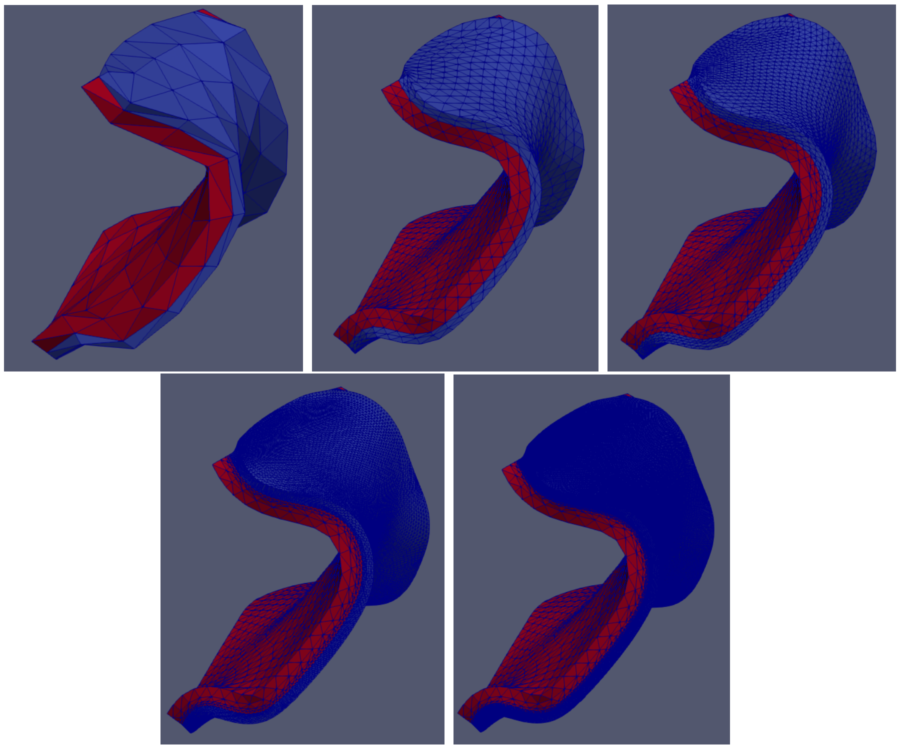

Figure 5.1: Mesh 0 (the coarsest), mesh 1, mesh 2, mesh 3, mesh 4 (the finest)

As in section 4.3.2, in the tables, BGS_S/G denotes the block Gauss-Seidel with SOR preconditioning the parabolic block and AMG preconditioning for the elliptic block. J_S/G denotes the block Jacobi with SOR preconditioning the parabolic block and AMG preconditioning for the elliptic block. BGS_G/G denotes the block Gauss-Seidel with AMG preconditioning for both of the parabolic and the elliptic block. AMG denotes AMG applied to the full bidomain system as the preconditioner. I performed numerical experiment with $\Delta t = 0.125$ ms and $\Delta t = 0.0625$ ms. Table (5.1) shows the results with conductivities obtained from Clerc [66] denoted as σ_1, for which

$$\sigma_{\mathrm{i}} = \begin{bmatrix} 1.7 & 0 & 0 \\ 0 & 0.19 & 0 \\ 0 & 0 & 0.19 \end{bmatrix}, \qquad \sigma_{\mathrm{e}} = \begin{bmatrix} 6.2 & 0 & 0 \\ 0 & 2.4 & 0 \\ 0 & 0 & 2.4 \end{bmatrix}, \qquad \sigma_{\mathrm{b}} = 20, \tag{5.1.1}$$

the units is mScm^{-1}.

Δt=0.125	BGS_S/G	J_S/G	BGS_G/G	AMG
mesh0	6	7	6	8
mesh1	9	9	9	15
mesh2	12	13	11	23
mesh3	15	15	13	33
mesh4	17	17	16	59
Δt=0.0625	**BGS_S/G**	**J_S/G**	**BGS_G/G**	**AMG**
mesh0	5	5	5	7
mesh1	8	8	8	15
mesh2	11	11	10	25
mesh3	14	15	12	34
mesh4	17	17	15	58

Table 5.1: Convergence iterations with σ_1.

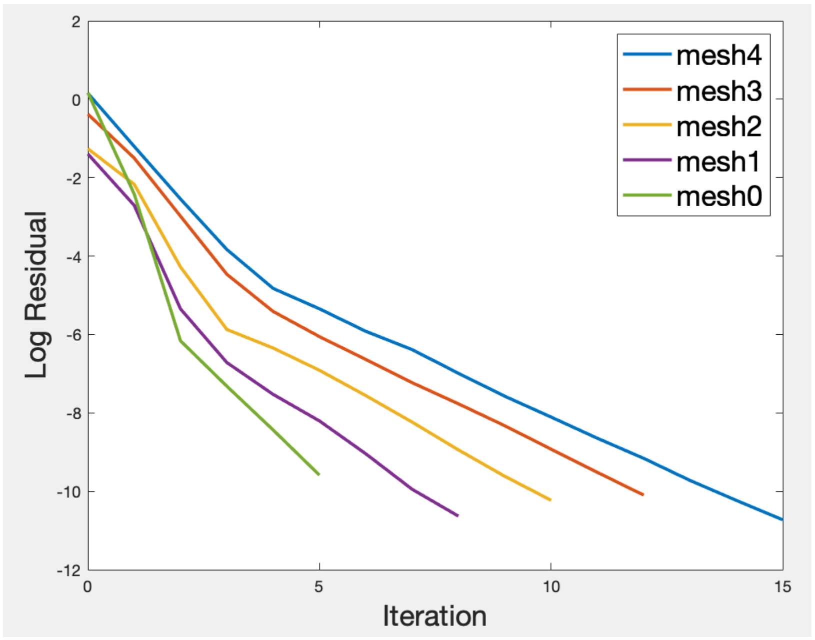

Figure 5.2: Iteration plot for BGS_G/G and σ_1, with $\Delta t = 0.0625$ ms.

According to the table and figure, the convergence rates for the same solver with the same mesh and different time-steps do not differ significantly. Block Gauss-Seidel with AMG for both blocks gives the smallest iterations. The results for the block methods differ little, while AMG preconditioner requires significantly more iterations to converge. Thus I only use Block Gauss-Seidel with AMG for the two blocks in the other simulations. Table (5.2) and figure (5.4) show the results with conductivities obtained from Graham et al. [67], denoted as σ_2, for which

$$\sigma_{\mathrm{i}} = \begin{bmatrix} 2.8 & 0 & 0 \\ 0 & 0.26 & 0 \\ 0 & 0 & 0.26 \end{bmatrix}, \qquad \sigma_{\mathrm{e}} = \begin{bmatrix} 2.2 & 0 & 0 \\ 0 & 1.3 & 0 \\ 0 & 0 & 1.3 \end{bmatrix}, \qquad \sigma_{\mathrm{b}} = 20, \tag{5.1.2}$$

the units is mScm^{-1}.

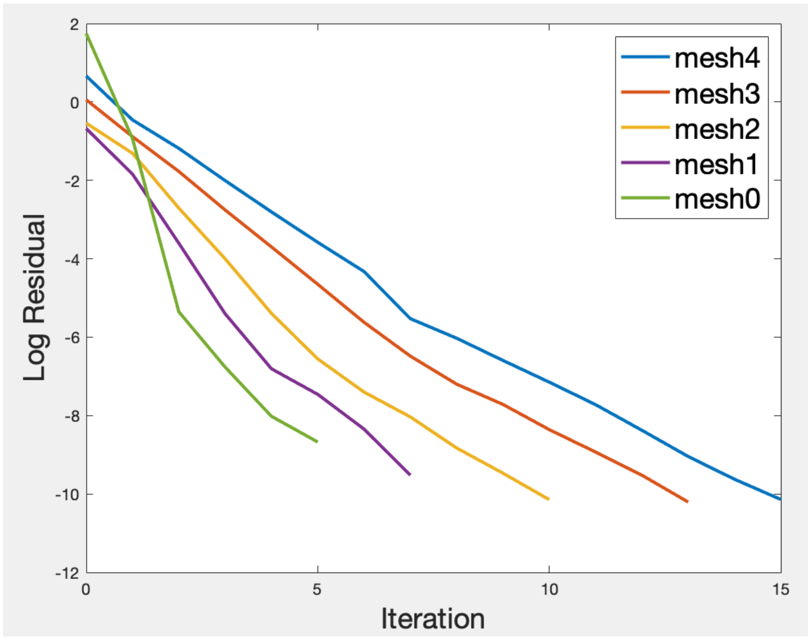

Figure 5.3: Iteration plot for BGS_G/G and σ_2, with $\Delta t = 0.0625$ ms.

	$\Delta t = 0.125$ ms	$\Delta t = 0.0625$ ms
mesh 0	6	5
mesh 1	8	7
mesh 2	10	10
mesh 3	13	13
mesh 4	16	15

Table 5.2: Convergence iterations with σ_2.

The convergence rates are almost similar for σ_1 and σ_2 with block Gauss-Seidel with AMG preconditioning for both blocks. For each finer mesh, the convergence iterations only increase 2 or 3 compared with the coarser mesh.

I also test for fibrosis case. Similar as in chapter 4, I tested 30%, 70%, and 90% fibrosis, with σ_1 and different time-step. The results are shown in table (5.3) and (5.4).

$\Delta t = 0.125$ ms	30%	70%	90%
mesh 0	6	6	7
mesh 1	7	7	8
mesh 2	11	11	12
mesh 3	13	13	13
mesh 4	16	16	15

Table 5.3: Convergence iterations for $30\%, 70\%$, and 90% fibrosis with BGS_G/G, $\Delta t = 0.125$ ms, and σ_1.

$\Delta t = 0.0625$ ms	30%	70%	90%
mesh 0	6	5	5
mesh 1	7	6	6
mesh 2	10	9	9
mesh 3	13	12	11
mesh 4	15	14	14

Table 5.4: Convergence iterations for $30\%, 70\%$, and 90% fibrosis with BGS_G/G, $\Delta t = 0.0625$ ms, and σ_1.

The convergence rates for the with fibrosis cases I tested are smaller compared with the without fibrosis cases in table (5.1), which is similar as in the fibrosis tests in section 4.3 and section 4.4. The 90% fibrosis solution plot is shown in figure (5.5).

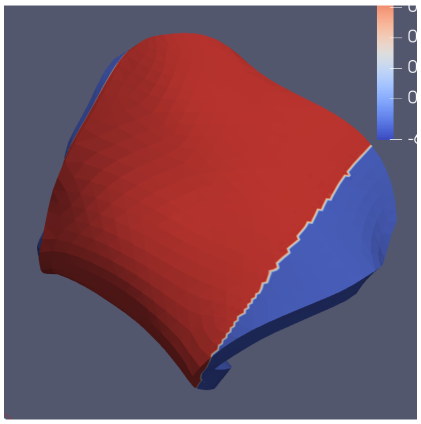

Figure 5.4: Solution plot for 90% fibrosis at $t = 1$ ms

5.2 Conclusion

In this section, I tested the AMG solver and the block factorization approach. As in the simple 2D geometries problems, the AMG solver performs poorly for solving the bidomain system, while the block factorization shows relatively good performance with only a mild increase in the number of iterations under grid refinement. In addition, the convergence iterations with block Jacobi and block Gauss-seidel do not differ significantly, for both the 2D and 3D tests in chapter 4 and 5.

Since AMG is developed based on the assumption that for $M\mathbf{x} = \mathbf{b}$, M should be SPD, as mentioned in chapter 3. For M not in the SPD form, we need to find other way to efficiently solve the linear system. The purpose of this thesis is to show that we can decompose M into blocks, with each block resemble the Poisson-type problem, which is SPD. Then we can apply AMG as the preconditioner for each block. According to the simulation results, this approach is efficient for solving the bidomain equations, both with and without bath.

CHAPTER 6

Conclusion

In this dissertation, I have:

1. showed that for idealized 2D geometry, block factorization with GMG preconditioner and GMG as a precondiitoner for GMRES are efficient methods for solving the bidomain equations (Chapter 4); and

2. showed that for idealized 2D geometry, block factorization with AMG preconditioner is an efficient method for solving the bidomain equations (Chapter 4); and

3. showed that for 3D geometry representing the realistic whole posterior wall of the left atrium, block factorization is an efficient method for solving the bidomain equations (Chapter 5); .

For the idealized geometries, I considered both bidomain equations with and without bath. The realistic geometry contains both the myocardium and the bath. For all three cases listed above, I considered different conductivities, as well as different time-step sizes. From the simulation results, convergence iterations with $\Delta t = 0.0125$ ms for the tests on the simple 2D geometries with block factorization approach increase about 0-2 iterations as the grid doubles in both x and y directions for the two different sets of conductivities.

In the GMG section, I experimented from 32×32 grids until 512×512 grids. For both the with and without bath casess, the convergence iterations of GMG solver is 4-5 more than those of the GMG preconditioner with the same conductivity and grids at $\Delta t = 0.0125$ ms. The convergence iterations of the block factorization is 1-3 more than those of the GMG preconditioner, while is 2-4 fewer than those of the the GMG solver. For the block factorization and the GMG preconditioner, convergence iterations with time-step $\Delta t = 0.05$ ms are about 1-3 more than those with $\Delta t = 0.0125$ ms at the same grids. In addition, I considered the fibrosis case, which is the the expansion of extracellular matrix and the increasing of the number of fibroblasts in the myocardium. I considered 30%, 70%, and 90% fibrosis cases, with GMG preconditioner and block factorization. Convergence iterations of GMG preconditioner is about 1-2 fewer than those of the block factorization of the same grids with the same fibrosis percentage, both for the with and without bath casess. The convergence iterations of the 90% case are about 1-2 iterations smaller than the other cases with the same grids.

In the AMG section, I experimented from 32×32 grids until 1024×1024 grids. The convergence iterations of AMG direct solver are growing fast, while those of the block factorization approach increase about 0-2 as the grid doubles in x and y directions. I considered three block factorization approaches for the bidomain without bath case, and the convergence iterations are almost similar of the three methods. Similar as in the GMG section, for the two different conductivities and for both the with and without bath cases, the convergence iterations with $\Delta t = 0.0125$ ms increase 0-2 as the grid doubles in x and y directions. In addition, I considered the non-aligned anisotropy case, of which the ionic current does not propagate in the directions parallel to the grids. I tested with $22.5°$. In this case, convergence iterations increase at about 0-2 as the grid doubles in the x and y directions, for both the with and without bath casess. The convergence iterations of the non-aligned case are 1-2 fewer than those of the aligned case, when using the rectangular element and $\Delta t = 0.0125$ ms. I also considered the 30%, 70%, and 90% fibrosis. Similar as in the GMG section, the convergence iterations of 90% fibrosis are 1 smaller than the other cases with the same grids. Also, the convergence iterations of all the fibrosis cases are fewer than those without fibrosis tests.

Similar as in the AMG section, in the realistic geometry section, the convergence iterations with different block factorization approaches are quite similar, growing 2-3 iterations as the mesh becomes finer. The AMG solver still performs poorly. For the fibrosis tests, similar as the in the GMG and AMG section, the convergence iterations of 90% fibrosis is the smallest.

The main contribution of this thesis is to show that with the block factorization approach, of which the sub-blocks are preconditioned with multigrid methods, we can obtain a mild increase in the number of iterations under grid refinement for solving the bidomain equations. Since AMG and GMG perform well for Poisson-like problems, for non-Poisson-like problems, such as the bidomain equations, we can decompose the operator matrix into Poisson-like blocks using methods such as block Jacobi and block Gauss-seidel, and apply AMG or GMG preconditioner to each sub-block.

REFERENCES

[1] W. L. Briggs, V. E. Henson, and S. F. McCormick, *A Multigrid Tutorial, 2nd Edition.* SIAM, 2000.

[2] M. Arnsdorf, "Cardiac electrophysiology," *SIAM Journal on Scientific Computing*, 2019.

[3] G. J. Tortora and N. P. Anagnostakos, *Principles of Anatomy and Physiology.* New York : Harper and Row, 5 ed., 1987.

[4] J. E. Hall and C. Guyton, *Textbook of Medical Physiology, Chapter 6.* Saunders, 12 ed., 2015.

[5] B. Gumbiner, "Cell adhesion: the molecular basis of tissue architecture and morphogenesis," *Cell*, vol. 84, pp. 345–357, 1996.

[6] P. C. Franzonem, L. F. Pavarino, and S. Scacchi, *Mathematical Cardiac Electrophysiology*, vol. 13. Springer, Cham, 2014.

[7] R. H. Clayton, O. Bernus, E. Cherry, H. Dierckx, F. Fenton, L. Mirabella, A. Panfilov, F. Sachse, G. Seemann, and H. Zhang, "Models of cardiac tissue electrophysiology: Progress, challenges and open questions," *PROGRESS IN BIOPHYSICS and MOLECULAR BIOLOGY*, vol. 104, pp. 22–48, 2011.

[8] S. Gilbert, A. Benson, P. Li, and A. Holden, "Regional localisation of left ventricular sheet structure: integration with current models of cardiac fibre, sheet and band structure," *European Journal of Cardiothoracic Surgery*, vol. 32, pp. 231–249, 2007.

[9] M. Lockhart, E. Wirrig, A. Phelps, and A. Wessels, "Extracellular matrix and heart development," *Birth Defects Res A Clin Mol Teratol*, pp. 535–550, 2011.

[10] R. E. Klabunde, *Cardiovascular Physiology Concepts.* Lippincott Williams and Wilkins, 2 ed., 2012.

[11] A. Holden and H. Zhang, "Modelling propagation and re-entry in anisotropic and smooth heterogeneous cardiac tissue," *Journal of the Chemical Society Faraday Transactions*, vol. 89, pp. 2833–2837, 1993.

[12] R. Winslow, A. Varghese, D. Noble, C. Adlakha, and A. Hoythya, "Generation and propagation of ectopic beats induced by spatially localized naek pump inhibition in atrial network models," *Proceedings of the Royal Society of London*, vol. 254, pp. 55–61, 1993.

[13] G. Bub, A. Shrier, and L. Glass, "Spiral wave generation in heterogenous excitable media," *Physical Review Letters*, vol. 88, 2002.

[14] I. Waller and R. Kapral, "Spatial and temporal structure in systems of coupled nonlinear oscillators," *Physical Review A*, vol. 30, pp. 2014–2055, 1984.

[15] Y. Rudy, "Reentry-insights from theoretical simulations in a fixed pathway," *Journal of Cardiovascular Electrophysiology*, vol. 6, no. 294-312, 1995.

[16] H. J. Jongsma and R. Wilders, "Gap junctions in cardiovascular disease," *Circulation Research*, pp. 1193–1197, 2000.

[17] M. K. Sunil, "The electrical bidomain model: A review," *Scholars Academic Journal of Biosciences (SAJB)*, vol. 7, pp. 633–639, 2015.

[18] O. Darrigol, *Electrodynamics from Ampère to Einstein.* Oxford University Press, 2000.

[19] R. Gray, D. Mashburn, V. Sidorov, and J. Wikswo, "Quantification of transmembrane currents during action potential propagation in the heart," *Biophys J.*, vol. 104, no. 1, pp. 268–278, 2013.

[20] P. Pathmanathan, M. O. Bernabeu, R. Bordas, J. Cooper, A. Garny, J. Pitt-Francis, J. Whiteley, and D. Gavaghan, "A numerican guide to the solution of the bidomain equations of cardiac electrophysiology," *Progress in biophysics and molecular biology*, vol. 102, pp. 136–155, 2010.

[21] A. Bueno-Orovio, V. Perez-Garcia, and F. Fenton, "Spectral methods for partial differential equations in irregular domains: the spectral smoothed boundary method," *SIAM Journal on Scientific Computing*, vol. 28, pp. 886–900, 2006.

[22] S. Gandhi and B. J. Roth, "A numerical solution of the mechanical bidomain model.," *Comput Methods Biomech Biomed Engin*, vol. 19, no. 10, pp. 1099–106, 2016.

[23] M. Potse, B. Dubé, J. Richer, A. Vinet, and R. M. Gulrajani, "A comparison of monodomain and bidomain reaction-diffusion models for action potential propagation in the human heart," *IEEE Trans Biomed Eng*, vol. 53, pp. 2425–35, 2006.

[24] H. Saleheen and K. Ng, "A new three-dimensional finite-difference bidomain formulation for inhomogeneous anisotropic cardiac tissues," *IEEE Trans. Biomed. Eng.*, vol. 45, no. 1, p. 15, 1998.

[25] R. Khan and K. Ng, "Higher order finite difference modeling of cardiac propagation," in *IEEE International Conference on Bioinformatics and Biomedicine*, 2017.

[26] K. Sharma and B. J. Roth, "The mechanical bidomain model of cardiac muscle with curving fibers," *PhysicalBiology*, vol. 15, no. 6, 2018.

[27] G. Huiskamp, "Simulation of depolarization in a membraneequations-based model of the anisotropic ventricle," *IEEE Trans Biomed Eng*, vol. 45, no. 7, 1998.

[28] S. M. Shuaiby, M. A. Hassan, and M. El-Melegy, "Modeling and simulation of the action potential in human cardiac tissues using finite element method," *J. of Commun. Comput. Eng*, vol. 2, no. 21-27, 2012.

[29] O. Steinbath and H. Yang, "A space–time finite element method for the linear bidomain equations," in *Lecture Notes in Computational Science and Engineering*, vol. 128, 2019.

[30] G. Seemann, F. Sachse, K. Karl, D. Weiss, V. Heuveline, and O. F. Dossel, "Framework for modular, flexible and efficient solving the cardiac bidomain equation using petsc," *Progress in Industrial Mathematics at ECMI*, 2008.

[31] H. Dal, S. Goktepe, M. Kaliske, and E. Kuhl, "A fully implicit finite element method for bidomain models of cardiac electromechanics," *Computer Methods in Applied Mechanics and Engineering*, vol. 253, pp. 323–336, 2013.

[32] E. Vigmond, M. Hughes, G. Plank, and L. Leon, "Computational tools for modeling electrical activity in cardiac tissue," *J. Electrocardiol*, vol. 3, pp. 69–74, 2003.

[33] M. C. Costa, F. O. Campos, A. Prassl, R. Weber dos Santos, D. Sanchez-Quintana, H. Ahammer, E. Hofer, and G. Plank, “An efficient finite element approach for modeling fibrotic clefts in the heart,” *IEEE Trans. Biomed. Eng.*, vol. 61, 2014.

[34] M. Buist, G. Sands, P. Hunter, and A. Pullan, “A deformable finite element derived finite difference method for cardiac activation problems,” *Annals of Biomedical Engineering*, vol. 31, pp. 577–588, 2003.

[35] M. Trew, L. L. Grice, B. S. Smaill, and A. Pullan, “A finite volume method for modeling discontinuous electrical activation in cardiac tissue,” *Annals of Biomedical Engineering*, vol. 33, pp. 590–602, 2005.

[36] Y. Courdiere and C. Pierre, “Stability and convergence of a finite volume method for two systems of reaction-diffusion equations in electro-cardiology,” *Nonlinear Analysis: Real World Applications*, vol. 7, pp. 916–935, 2006.

[37] R. Johnston, “A finite volume method solution for the bidomain equations and their application to modelling cardiac ischaemia,” *Computer Methods in Biomechanics and Biomedical Engineering*, vol. 13, pp. 157–170, 2010.

[38] R. C. Penland, D. M. Harrild, and C. S. Henriquez, “Modeling impulse propagation and extracellular potential distributions in anisotropic cardiac tissue using a finite volume element discretization,” *Computing and Visualization in Science*, vol. 4, pp. 215–226, 2002.

[39] S. Puwal and B. J. Roth, “Forward euler stability of the bidomain model of cardiac tissue,” *IEEE Trans. Biomed. Eng.*, vol. 54, no. 5, 2007.

[40] A. L. Muzikant, E. W. Hsu, P. D. Wolf, and C. S. Henriquez, “Region specific modeling of cardiac muscle: Comparison of simulated and experimental potentials,” *Annals of Biomedical Engineering*, vol. 30, pp. 867–883, 2002.

[41] R. Weber dos Santos, G. Plank, S. Bauer, and E. J. Vigmond, “Parallel multigrid preconditioner for the cardiac bidomain model,” *IEEE Trans Biomed Eng*, vol. 51, pp. 1960–8, 2004.

[42] E. Cherry, H. S. Greenside, and C. S. Henriquez, “Efficient simulation of three-dimensional anisotropic cardiac tissue using an adaptive mesh refinement method,” *Chaos*, vol. 13, no. 3, pp. 853–65, 2003.

[43] M. Ethier and Y. Bourgault, “Semi-implicit time-discretization schemes for the bidomain model,” *SIAM Journal on Numerical Analysis*, vol. 46, pp. 2443–2468, 2008.

[44] M. Murillo and X. Cai, “A fully implicit parallel algorithm for simulating the non-linear electrical activity of the heart,” *Numer. Linear Algebra Appl.*, vol. 11, no. 11, pp. 261–277, 2004.

[45] Y. Bourgault, M. Ethier, and V. G. LeBlanc, “Simulation of electrophysiological waves with an unstructured finite element method,” *Math. Model. Numer. Anal.*, vol. 37, pp. 649–661, 2003.

[46] M. Munteanu and L. F. Pavarino, “Implicit parallel solvers in computational electrocardiology,” *Applied Analysis and Differential Equations*, 2007.

[47] M. Pennacchio and V. Simoncini, “Efficient algebraic solution of reaction-diffusion systems for the cardiac excitation process,” *J. Comput. Appl. Math.*, vol. 145, pp. 49–70, 2002.

[48] P. C. Franzonem and L. F. Pavarino, "A parallel solver for reaction-diffusion systems in computational electrocardiology," *Math. Models Methods Appl. Sci.*, vol. 14, pp. 883–912, 2004.

[49] J. Whiteley, "An efficient numerical technique for the solution of the monodomain and bidomain equations," *IEEE Trans. Biomed. Eng*, vol. 53, no. 11, pp. 2139–2147, 2006.

[50] J. Whiteley, "Physiology driven adaptivity for the numerical solution of the bidomain equations," *Annals of Biomedical Engineering*, vol. 35, no. 9, pp. 1510–1520, 2007.

[51] J. Whiteley, "An efficient numerical technique for the solution of the monodomain and bidomain equations," *IEEE TRANSACTIONS ON BIOMEDICAL ENGINEERING*, vol. 53, no. 11, 2006.

[52] G. T. Lines, M. Buist, P. Grottum, A. J. Pullan, J. Sundnes, and A. Tveito, "Mathematical models and numerical methods for the forward problem in cardiac electrophysiology," *Computing and Visualization in Science*, vol. 5, no. 215-239, 2003.

[53] J. Sundnes, G. Lines, and A. Tveito, "Efficient solution of o rdinary differential equations modeling electrical activity in cardiac cells," *Mathematical Biosciences*, vol. 172, pp. 55–72, 2001.

[54] J. C. Eason and R. A. Malkin, "A simulation study evaluating the performance of high-density electrode arrays on myocardial tissue," *IEEE TRANSACTIONS ON BIOMEDICAL ENGINEERING*, vol. 47, no. 7, 2000.

[55] E. Vigmond, R. Weber dos Santos, A. Prassl, M. Deo, and G. Plank, "Solvers for the cardiac bidomain equations," *Prog Biophys Mol Biol*, vol. 96, pp. 3–18, 2008.

[56] L. Gerardo-Giorda, L. Mirabella, F. Noble, M. Perego, and A. Veneziani, "A model-based block-triangular preconditioner for the bidomain system in electrocardiology," *Journal of Computational Physics*, vol. 228, pp. 3625–3639, 2009.

[57] W. Santos, *Modelling cardiac electrophysiology.* PhD thesis, Federal University of Rio de Janeiro, 2002.

[58] J. P. Keener and K. Bogar, "A numerical method for the solution of the bidomain equations in cardiac tissue," *Chaos*, vol. 8, 1998.

[59] T. Austin, M. Trew, and A. Pullan, "Solving the cardiac bidomain equations for discontinuous conductivities," *IEEE Trans. Biomed. Eng*, vol. 53, no. 7, pp. 1265–1272, 2006.

[60] J. Sundnes, G. T. Lines, K. A. Mardal, and A. Tveito, "Multigrid block preconditioning for a coupled system of partial differential equations modeling the electrical activity in the heart," *Comput Methods Biomech Biomed Engin*, vol. 5, pp. 397–409, 2002.

[61] R. Weber dos Santos, G. Plank, S. Bauer, and E. J. Vigmond, "Preconditioning techniques for the bidomain equations," *Lecture Notes In Computational Science And Engineering*, vol. 40, no. 571-580, 2004.

[62] T. Austin, M. Trew, and A. Pullan, "A comparison of multilevel solvers for the cardiac bidomain equations," *Conf Proc IEEE Eng Med Biol Soc*, vol. 7, pp. 7204–7, 2005.

[63] G. Plank, R. Weber dos Santos, E. Vigmond, and G. Haase, "Algebraic multigrid preconditioner for the cardiac bidomain model," *IEEE Trans. Biomed. Eng.*, vol. 54, no. 4, pp. 585–596, 2007.

[64] M. Pennacchio and V. Simoncini, "Algebraic multigrid preconditioners for the bidomain reaction–diffusion system," *Applied Numerical Mathematics*, vol. 59, pp. 3033–3050, 2009.

[65] M. Pennacchio and V. Simoncini, "Fast structured amg preconditioning for the bidomain model in electrocardiology," *SIAM Journal on Scientific Computing*, vol. 33, pp. 721–745, 2011.

[66] Clerc, "Directional differences of impulse spread in trabecular muscle from mammalian heart," *J. Physiol. (Lond.)*, vol. 225, no. 2, pp. 335–346, 1976.

[67] L. S. Graham, D. Kilpatrick, F. Sainsbury, and A. C. Yong, "The determination of the bidomain conductivity values of heart tissue," *Computers in Cardiology*, vol. 35, pp. 193–196, 2008.

[68] Y. Gao, Y. Gong, and L. Xia, "Simulation of atrial fibrosis using coupled myocyte-fibroblast cellular and human atrial models," *Computational and Mathematical Methods in Medicine*, 2017.

[69] T. P. Nguyen, Z. Qu, and J. N. Weiss, "Cardiac fibrosis and arrhythmogenesis: The road to repair is paved with perils," *J Mol Cell Cardiol*, vol. 0, pp. 83–91, 2014.

[70] S. Rossi, S. Gaeta, B. Griffith, and C. S. Henriquez, "Muscle thickness and curvature influence atrial conduction velocities," *Front. Physiol.*, 2018.