A MULTILEVEL ADAPTIVE APPROACH FOR COMPUTATIONAL CARDIOLOGY

by

Wenjun Ying

Department of Mathematics
Duke University

Date: __________________________

Approved:

Dr. John A. Trangenstein, Supervisor

Dr. David G. Schaeffer

Dr. William K. Allard

Dr. Craig S. Henriquez

Dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Department of Mathematics in the Graduate School of Duke University

2005
ABSTRACT
(MATHEMATICS)

A MULTILEVEL ADAPTIVE APPROACH FOR COMPUTATIONAL CARDIOLOGY

by

Wenjun Ying

Department of Mathematics
Duke University

Date: _____________________

Approved:

________________________________
Dr. John A. Trangenstein, Supervisor

________________________________
Dr. David G. Schaeffer

________________________________
Dr. William K. Allard

________________________________
Dr. Craig S. Henriquez

An abstract of a dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Department of Mathematics in the Graduate School of Duke University 2005
Abstract

A space-time adaptive mesh refinement (AMR) algorithm is proposed for cardiac problems, arising from numerical modeling of electrical wave propagation in the heart. A cardiac problem, such as the bidomain or monodomain model, consists of a singularly perturbed reaction-diffusion system coupled with a set of nonlinear stiff ordinary differential equations. The algorithm first uses an operator splitting technique to separate linear diffusion from nonlinear stiff reactions in the model problem to be solved. The reactions are integrated adaptively with a second-order singly diagonally implicit Runge-Kutta method. The decoupled linear diffusion is implicitly discretized with a conforming finite element approximation on adaptively refined grids, which are dynamically created by the AMR algorithm. The resulting composite grid equations are solved by a standard multilevel/multigrid iteration algorithm.

The AMR algorithm uses quadrilateral or hexahedral elements to construct a hierarchy of properly nested level grids. The grid on each level is generated through regular bisection or refinement from a subgrid of that on the previous coarser level. We represent a grid by lists of elements and the lower dimensional facets, including nodes, edges and sides. With this grid representation, the connectivity of grid entities is well established, which makes grid-based operations easy to implement. Since the composite grids in the hierarchy do not match along coarse-fine grid interfaces, the Steklov-Poincare continuity conditions are weakly enforced, which results in a conforming finite element discretization on composite grids.

The AMR algorithm follows Berger-Oliger’s approach in time stepping. The hierarchical system is recursively integrated. Starting from the root level, a coarse level is first integrated with a large time step, followed by integrating its next finer level with a few small time steps until both levels are synchronized. At the moment
of synchronization, the algorithm invokes some of the typical routines such as mesh regridding and data up-/downscaling. An error estimation technique of Richardson extrapolation type is proposed for tagging of cells.

Numerical experiments with the AMR algorithm are presented at the end as part of the thesis work. It is indicated that the convergence rate of our AMR algorithm is second-order. Numerical results also demonstrate the numerical accuracy, algorithm efficiency, geometry flexibility of our AMR algorithm and its promising application to realistic simulations.
Acknowledgements

Many individuals over the years deserve acknowledgement in this dissertation. First of all, I am greatly indebted to Professor John A. Trangenstein, who suggested, supervised the thesis work and very carefully proofread the dissertation. I would like to express my utmost gratitude and heartfelt appreciation to him for his time, patience, support, guidance, generosity and his friendship, which have proved invaluable to me over the last five years. I am also extremely grateful to Professor Dave G. Schaeffer for his always encouragement and forever kindness. I enjoy the work with Professor Schaeffer very much. Many thanks go to Professor Bill K. Allard for his time and efforts with me on distributed computing (Due to the limitation of space and time, this dissertation does not include the wonderful work with Professor Allard on parallel computing). I also acknowledge Professor Thomas J. Beale for the long time studies from him.

I would like to thank Professor Craig S. Henriquez for all of the time and discussions with him. Without his guidance, help and support, the algorithm proposed in the thesis work for computational cardiology would never become useful, realize its importance and be further improved. I also appreciate very much the opportunity I have had to work with Professor Wanda Krassowska. I have learned from both of them to appreciate the value of understanding the viewpoints that researchers in different disciplines have with regard to common research problems.

Special thanks go to Dr. Xabier Garaizar and Dr. Rich Hornung in the center for applied scientific computing of the Lawrence Livermore National Laboratory for their hospitality during my stay there. By the way, I would like to again thank our “car driver”, Professor John A. Trangenstein. Without his help, we would never have the weekend trips to visit Yosemite, the Golden Gate Bridge and the redwood parks
in the bay area of San Francisco.

I am also grateful to Dr. Elizabeth M. Cherry for sharing of the bibliography of her dissertation. This saves me days and nights of time in the thesis writing.

I would like to thank all of the faculty and the staff of the Department of Mathematics at Duke University for their wonderful help. Special thanks go to Professor Lewis Blake and Professor Clark B. Bray for improving my teaching skills. I really appreciate the opportunity that they gave me to teach an undergraduate calculus class. I am also very much indebted to our computer administers, Dr. Yunliang Yu and Andrew R. Schretter. They are always my life saviors, especially when I accidentally deleted all of the code lines I wrote for the thesis.

I also would like to thank my officemates and friends here at Duke for their friendship, kindness and help. Especially I want to thank John Cain, Robert Karp, Guoqiang Yang, Ryan Deering, John Greer, Robert Buckingham, Daniel Fox, Joseph Spivey, Zhentian Xie and Chunsheng Liu etc.. I also would like to express my appreciation to the Chinese Community in the area.

I am extremely grateful to our PopPop and Nony, Edward and Ann Rocap. It is them who helped our family so much during the pregnancy of my wife and the growing of our daughter that I was able to be fully devoted to the thesis work. It is them who make us feel not lonely on the holidays even though we are far from our home country.

I also thank my former adviser, Professor Houde Han, back in my home university for his support and encouragement.

Finally, this dissertation is dedicated to my parents, brother and sister in China and my family here in the States.
To

My former adviser

Professor Houde Han

in Tsinghua University
# Contents

Abstract iv

Acknowledgements vi

List of Tables xv

List of Figures xix

## 1 Introduction 1

1.1 Computational Cardiology ........................................ 1
   1.1.1 Action Potentials ............................................ 3
   1.1.2 Computational Issues ........................................ 5

1.2 Literature Review ................................................ 8
   1.2.1 Numerical Methods for Cardiac Problems ...................... 8
   1.2.2 Adaptive Methods for General Problems ...................... 13

1.3 Thesis Objectives ................................................. 17

1.4 Outline of Thesis ................................................ 20

## 2 Model Problems 25

2.1 The Bidomain Model ............................................... 25
   2.1.1 Introduction ................................................... 28
   2.1.2 Homogenization ............................................... 32
   2.1.3 Anisotropic Conductivity .................................... 38

2.2 The Simplified Monodomain Model ................................ 40

2.3 Cardiac Models for Membrane Dynamics .......................... 41

2.4 Reaction-Diffusion Systems ........................................ 44
4.4 Elements of the Adaptive Process ........................................... 109
  4.4.1 Recursive Integration .................................................. 109
  4.4.2 Timestep Selection .................................................... 109
  4.4.3 Regridding of Fine Levels ........................................... 111
  4.4.4 Tagging of Coarse Cells .............................................. 112
  4.4.5 Data Up/Down Scaling ............................................... 115
  4.4.6 Interface Data Correction .......................................... 116

4.5 Initial and Boundary Conditions ....................................... 117

4.6 Error Estimation ............................................................ 119

5 Multilevel/Multigrid Iterations ......................................... 124
  5.1 Introduction .................................................................... 124
  5.2 Subspace Correction Methods ......................................... 126
  5.3 The Standard V-cycle Multigrid Algorithm ......................... 133
  5.4 Iteration on Uniformly Refined Grids ............................... 146
    5.4.1 Smoothing/Relaxation ............................................... 147
    5.4.2 Coarse Grid Correction ............................................ 149
  5.5 Iteration on Locally Refined Grids .................................... 150
    5.5.1 Smoothing/Relaxation ............................................... 154
    5.5.2 Coarse Grid Correction ............................................ 157
  5.6 Implementation Details .................................................. 158
    5.6.1 Initial Guess ......................................................... 158
    5.6.2 Coarsest Level Solver .............................................. 160
    5.6.3 Stopping Criterion .................................................. 160
6 Numerical Experiments 166

6.1 Introduction .................................................. 166
6.2 Convergence Rate ................................................. 170
   6.2.1 The Case in One Space Dimension ....................... 171
   6.2.2 The Cases in Two Space Dimensions ..................... 174
6.3 Numerical Accuracy ............................................. 179
6.4 Algorithm Efficiency ........................................... 181
   6.4.1 The Case in One Space Dimension ....................... 181
   6.4.2 The Case in Two Space Dimensions ..................... 184
   6.4.3 The Case in Three Space Dimensions ................... 186
6.5 Geometry Flexibility ......................................... 188
6.6 Applications .................................................. 188

7 Conclusion and Future Work 206

7.1 Thesis Achievements .......................................... 206
7.2 Future Work .................................................. 207

A Results by Space Only Adaptive Integration 210

A.1 The Case in One Space Dimension .......................... 210
A.2 The Cases in Two Space Dimensions ..................... 211

B Results with Exact Error Based Tagging of Cells 214

B.1 The Case in One Space Dimension .......................... 215
B.2 The Cases in Two Space Dimensions ..................... 216

C Results with Larger Tolerance for Tagging of Cells 220

C.1 The Case in One Space Dimension .......................... 221
C.2 The Cases in Two Space Dimensions .......................... 222

D Spiral Wave Results of the FitzHugh-Nagumo Model in 2D 226

E Scroll Wave Results of the FitzHugh-Nagumo Model in 3D 232

F AMR for the Hyperbolic Conservation Laws of Gas Dynamics 238

Bibliography 248

Biography 266
List of Tables

6.1 AMR for the 1D test problem at time $t = 0.25$ \ldots 171
6.2 AMR for the 1D test problem at time $t = 0.5$ \ldots 171
6.3 AMR for the 1D test problem at time $t = 1$ \ldots 171
6.4 Location of the 1D traveling wave front at $t = 1$ \ldots 173
6.5 AMR for the first 2D test problem at time $t = 0.25$ \ldots 176
6.6 AMR for the first 2D test problem at time $t = 0.5$ \ldots 176
6.7 AMR for the first 2D test problem at time $t = 1$ \ldots 176
6.8 AMR for the second 2D test problem at time $t = 0.25$ \ldots 202
6.9 AMR for the second 2D test problem at time $t = 0.5$ \ldots 202
6.10 AMR for the second 2D test problem at time $t = 1$ \ldots 202
6.11 AMR for the second 2D test problem at time $t = 2$ \ldots 202
6.12 Times of AMR for the 1D FitzHugh-Nagumo problem ($tol_{AMR} = 0.1$) \ldots 203
6.13 Times of uniform refinement for the 1D FitzHugh-Nagumo problem \ldots 203
6.14 Times of AMR for the 1D FitzHugh-Nagumo problem ($tol_{AMR} = 0.02$) \ldots 204
6.15 Times of AMR for the 1D FitzHugh-Nagumo problem ($tol_{AMR} = 0.05$) \ldots 204
6.16 Times of AMR for the 2D FitzHugh-Nagumo problem ($tol_{AMR} = 0.1$) \ldots 205
6.17 Times of uniform refinement for the 2D FitzHugh-Nagumo problem \ldots 205
A.1 Space Only AMR for the 1D test at time $t = 0.25$ \ldots 211
A.2 Space Only AMR for the 1D test at time $t = 0.5$ \ldots 211
A.3 Space Only for the 1D test at time $t = 1$ . . . . . . . . . . 211
A.4 Space Only AMR for the first 2D test at time $t = 0.25$ . . . . . . . . 212
A.5 Space Only AMR for the first 2D test at time $t = 0.5$ . . . . . . . . 212
A.6 Space Only AMR for the first 2D test at time $t = 1$ . . . . . . . . 212
A.7 Space Only AMR for the second 2D test at time $t = 0.25$ . . . . . . . 213
A.8 Space Only AMR for the second 2D test at time $t = 0.5$ . . . . . . . 213
A.9 Space Only AMR for the second 2D test at time $t = 1$ . . . . . . . 213
A.10 Space Only AMR for the second 2D test at time $t = 2$ . . . . . . . 213
B.1 Space Only AMR for the 1D test at time $t = 0.25$ . . . . . . . . . 215
B.2 Space Only AMR for the 1D test at time $t = 0.5$ . . . . . . . . . 215
B.3 Space Only for the 1D test at time $t = 1$ . . . . . . . . . . . . . 215
B.4 Space Only AMR for the first 2D test at time $t = 0.25$ . . . . . . . 216
B.5 Space Only AMR for the first 2D test at time $t = 0.5$ . . . . . . . 216
B.6 Space Only AMR for the first 2D test at time $t = 1$ . . . . . . . 216
B.7 Space Only AMR for the second 2D test at time $t = 0.25$ . . . . . . . 218
B.8 Space Only AMR for the second 2D test at time $t = 0.5$ . . . . . . . 218
B.9 Space Only AMR for the second 2D test at time $t = 1$ . . . . . . . 218
B.10 Space Only AMR for the second 2D test at time $t = 2$ . . . . . . . 218
C.1 Space Only AMR for the 1D test at time $t = 0.25$ . . . . . . . . . 221
C.2 Space Only AMR for the 1D test at time $t = 0.5$ . . . . . . . . . 221
C.3  Space Only for the 1D test at time $t = 1$  \ldots  \ldots  \ldots  \ldots  \ldots  \ldots  \ldots  \ldots  \ldots  221

C.4  Space Only AMR for the first 2D test at time $t = 0.25$  \ldots  \ldots  \ldots  \ldots  \ldots  \ldots  \ldots  \ldots  \ldots  222

C.5  Space Only AMR for the first 2D test at time $t = 0.5$  \ldots  \ldots  \ldots  \ldots  \ldots  \ldots  \ldots  \ldots  \ldots  222

C.6  Space Only AMR for the first 2D test at time $t = 1$  \ldots  \ldots  \ldots  \ldots  \ldots  \ldots  \ldots  \ldots  \ldots  222

C.7  Space Only AMR for the second 2D test at time $t = 0.25$  \ldots  \ldots  \ldots  \ldots  \ldots  \ldots  \ldots  \ldots  \ldots  224

C.8  Space Only AMR for the second 2D test at time $t = 0.5$  \ldots  \ldots  \ldots  \ldots  \ldots  \ldots  \ldots  \ldots  \ldots  224

C.9  Space Only AMR for the second 2D test at time $t = 1$  \ldots  \ldots  \ldots  \ldots  \ldots  \ldots  \ldots  \ldots  \ldots  224

C.10 Space Only AMR for the second 2D test at time $t = 2$  \ldots  \ldots  \ldots  \ldots  \ldots  \ldots  \ldots  \ldots  \ldots  224
### List of Figures

1.1 Cardiac action potential by the Beeler-Reuter model ................................ 3
1.2 Cardiac action potential by the Luo-Rudy phase one model ....................... 4
1.3 Cardiac action potential by the Luo-Rudy dynamic model ......................... 4
1.4 Issues from computational electrophysiology ........................................... 9
2.1 Intracellular and extracellular spaces $\Omega_i$ and $\Omega_e$ (4 rectangular cells) 29
2.2 Portions of intra- and extracellular spaces $\Omega_i$ and $\Omega_e$ within a unit cell 32
3.1 Richardson extrapolation ........................................................................... 64
3.2 Hanging nodes $A$ and $B$ on a composite grid from local refinement .......... 69
3.3 2D bijective mapping $J : \hat{T} \rightarrow T$ ..................................................... 70
3.4 3D bijective mapping $J : \hat{T} \rightarrow T$ ..................................................... 71
3.5 A computational domain consisting of two disjoint subdomains ............... 78
3.6 A two-level composite grid ..................................................................... 84
4.1 Hierarchical ordering of cells by a Morton space filling curve ................. 91
4.2 Hierarchical ordering of cells by a Hilbert-Peano space filling curve ....... 92
4.3 Morton space filling curves ...................................................................... 93
4.4 Hilbert-Peano space filling curves ........................................................... 94
4.5 Hierarchical ordering of lower dimensional grid entities (cells are ordered by a Morton space filling curve) ......................................................... 97
4.6 Hierarchical ordering of lower dimensional grid entities (cells are ordered by a Hilbert-Peano space filling curve) ......................................... 99
6.3 Adaptively refined grids and solution iso-contours (0.0001 – 0.9999) for the second 2D test problem at time \( t = 2 \) .................. 178

6.4 Action potential of the 1D FitzHugh-Nagumo problem at time \( t = 2 \) (with uniform grids and ten levels of refinement: \( \Delta x = \Delta t = 1/4096 \)) 181

6.5 Membrane potentials of the 1D FitzHugh-Nagumo model at time \( t = 2 \) (with the space-time adaptive integration algorithm and \( tol_{AMR} = 0.1 \)) 182

6.6 Adaptively refined grids and iso-contours (0.01–0.99) of the membrane potentials for the 2D FitzHugh-Nagumo model ................. 187

6.7 Adaptively refined grids and potential iso-surfaces (\( v = 0.5 \)) for the 3D FitzHugh-Nagumo problem .............................. 190

6.8 The FitzHugh-Nagumo monodomain model on complex domain (2D) 192

6.9 Membrane potential of the Luo-Rudy phase one model in one space dimension (data are collected at \( x = 0.5 \text{ cm} \)) ................. 194

6.10 Membrane potential of the Beeler-Reuter model in one space dimension (data are collected at \( x = 1.0 \text{ cm} \)) .................. 195

6.11 Luo-Rudy phase one bidomain model: transmembrane and extracellular potentials ............................................. 200

D.1 The FitzHugh-Nagumo monodomain model in two space dimensions . 230

E.1 The FitzHugh-Nagumo monodomain model in three space dimensions 236

F.1 The wind tunnel with a step 2D test problem for gas dynamics ....... 243

F.2 The oblique double Mach reflection on a 30° wedge .................. 246
Chapter 1

Introduction

The thesis work will focus on the application of an adaptive mesh refinement (AMR) algorithm to cardiac problems arising from modeling the electrical activity of the heart.

In this chapter we first introduce some background about computational cardiology and summarize some computational issues associated in section 1.1. In section 1.2, we make a literature review on numerical methods for the cardiac problems. In the same section, we also present a literature review on adaptive methods for general problems. Before going on to the details of cardiac models and the description of numerical methods, we list our strategies to the computational issues and the thesis objectives in section 1.3, which will guide the design of our numerical algorithm for the cardiac problems. Finally, we give an outline of the thesis in section 1.4.

1.1 Computational Cardiology

Computation of electrical wave propagation in the heart is one of the most important recent applications of mathematical modeling in biology. The main research direction of this subfield of biomedical engineering is the development of a realistic model of the whole mammalian heart. The realization of such a model has become feasible because of recent progress in nonlinear dynamics, and advance of computer technology, and experiments on cardiac tissue. For a comprehensive and practical introduction to modeling electrical activity of cardiac tissue and the whole heart, refer to the books [62, 108, 141].
The key aspect of the physiology of the heart is that it is a reliable, rhythmic pump that maintains the circulation throughout our life. Any interruption to its repetitive beating for more than a few minutes can lead to peripheral circulatory collapse and death. This mechanical activity is driven by the electrical activity of the heart, involving electrical waves propagating through cardiac tissue that trigger contraction in its muscle cells. The rhythmic pumping of the heart is produced by nearly synchronous contraction in the two atria, which pump blood into the ventricles, followed some hundred milliseconds afterwards by nearly synchronous contraction of the two ventricles that eject blood into the circulation. These two contractions, which form the cardiac cycle, are driven by rhythmic waves of electrical activity, propagating rapidly through the atria and then, after a delay, through the ventricles. This electrical activity is the excitation of the heart.

Modeling the electrical activity of the heart can help us understand the mechanisms responsible for the heart diseases such as atrial flutter, ventricular fibrillation and some types of paroxysmal tachycardia. It is believed that these disorders of heart functioning are concerned with abnormal spatial-temporal patterns of excitation in the cardiac muscle, in particular the re-entrant propagation of excitation through the same closed pathways in the tissue. Mathematical modeling can provide an important support and guide to a detailed analysis of spatial-temporal excitation patterns during the processes of generation and termination of the arrhythmias.

The electrical activity of the heart is modeled by a system of singularly perturbed reaction-diffusion partial differential equations coupled with a set of ordinary differential equations representing cell membrane dynamics (see Chapter 2 for details). It is possible to study the effects of various types of electric stimulation on cardiac arrhythmias by numerically solving the differential equations, while some technical difficulties associated with physical experiments are circumvented.
Figure 1.1: Cardiac action potential by the Beeler-Reuter model

However, there exist computational issues in the numerical simulation of electrical wave propagation in the heart. It is exactly our objective to tackle down the numerical difficulties and make the simulation more accurate, efficient and reliable.

Considering that some computational issues are closely related to a dynamic property of the cardiac tissue, we would like to describe it first before the difficulties are presented.

1.1.1 Action Potentials

The cardiac tissue is excitable in the sense that it responds to a small electrical perturbation by a local, small amplitude response, but if the perturbation is sufficiently large (above a threshold, in intensity and duration) the response is a large amplitude propagating electrical wave, called an action potential.

An action potential consists of five phases (see Figures 1.1-1.3): upstroke or depo-
Figure 1.2: Cardiac action potential by the Luo-Rudy phase one model

Figure 1.3: Cardiac action potential by the Luo-Rudy dynamic model
larization (phase 0), early repolarization (phase 1), plateau (phase 2), repolarization (phase 3) and resting (phase 4). During the action potential upstroke, the potential across the cell membrane changes from negative to positive. Immediately after an action potential upstroke, it is not possible to trigger a second action potential. Time must be allowed to elapse before a second one can be triggered. This phenomenon is known as refractoriness. The time during which it is not possible to trigger a second action potential is known as the refractory period. Immediately after the action potential upstroke and preceding the action potential plateau, there is a rapid phase of repolarization. The early rapid phase of repolarization is the result of the activation of transient outward current. After the phase of early repolarization is the plateau phase of the action potential. In this phase, repolarization is exceedingly slow and as a result the membrane potential remains at relatively positive potentials for several hundred milliseconds. The plateau phase is terminated by a more rapid phase of repolarization, which returns the membrane potential to the resting potential. Finally, the membrane potential stays at the stable resting state until the next action potential comes.

1.1.2 Computational Issues

For the simplicity of exposition, let us first introduce a general form of the differential equations modeling the electrical activity of the heart. As stated, a typical cardiac model consists of a system of singularly perturbed reaction-diffusion coupled with a set of stiff nonlinear ordinary differential equations, which has the following form:

\[
C_m \frac{\partial u}{\partial t} = \frac{1}{\beta} \nabla \cdot D(x) \nabla u - I_{ion}(u, q) \quad \text{for } x \in \Omega, \ t > 0, \quad (1.1a)
\]

\[
\frac{\partial q}{\partial t} = M(u, q) \quad \text{for } x \in \Omega, \ t > 0. \quad (1.1b)
\]
Here, $\Omega$ represents the computational domain defined by the heart; $x$ is the space variable; $t$ is the time variable; $u$ is a vector of cardiac potentials; $q$ is a vector of state variables, including the ionic concentrations and gating variables; $C_m$ is the membrane capacitance matrix; $D(x)$ is the spatially dependent conductivity tensor; $I_{\text{ion}}(u, q)$ and $M(u, q)$ are nonlinear functions, describing the cardiac membrane dynamics. For a specific cardiac model and its details, such as the bidomain or monodomain model, see the next chapter on model problems. Given reasonable boundary and initial conditions, the differential equations (1.1) can be numerically solved with a grid-based method, such as finite difference method, finite element method or finite volume method. It is well-recognized that the computational issues arising in the numerical simulation are associated with the symbols: $\Omega$, $x$, $t$, $I_{\text{ion}}$, $M$, $D$, $C_m$.

First, the computational domain $\Omega$ defined by the heart is geometrically complex. This makes accurate numerical simulation very difficult since it is hard for some numerical methods such as the finite difference method, to accurately approximate the complex boundary of the heart domain. We call this issue as **Complexity of Domain**.

Second, an action potential front spatially ($x$) extends only a few cardiac cells due to the finite speed of the traveling electrical wave and the fast upstroke of the cardiac action potential (see Figures 1.1-1.3). The computational grids used must be fine enough to get a physiologically realistic simulation of the cardiac electrical dynamics. This requires a spatial grid size of the order of the cardiac cell size. A whole heart simulation using a few grid points per cardiac cell or a few cells per grid element is well beyond the memory capacity of most modern computers [141]. For example, a whole heart simulation typically involves $O(10^7)$ unknowns [143]. We call this issue as **Spatial Locality of Electrical Wavefronts**.

Third, there is another issue related to the time variable $t$. As discussed in the
previous subsection, a typical cardiac action potential have very fast early repolariza-
tion (phase 1), exceedingly slow plateau (phase 2) and relatively slow repolarization
(phase 3). This means that multiple time scales are involved in an action potential.
The numerical methods taking into account only one scale or not choosing the time
step size small enough usually fail to give convergent and accurate results. That is,
the time step size may be severely restricted by the stability requirement. We call
this computational issue as **Multiscale Phenomena of Cardiac Excitation**, or
**Stiffness of Membrane Dynamics**.

The fourth issue is associated with the membrane capacitance $C_m$. In the case
that the cardiac problem represents the bidomain model, which will be introduced in
Chapter 2, the capacitance matrix $C_m$ is not invertible due to the current balance of
intra- and extracellular spaces. Explicit time integration methods, such as the forward
Euler method, can not be employed to solve the equations in this case. Instead an
implicit time integration must be applied instead. This needs an elliptic solver for
the resulting systems. However, the size of a typical system may be extremely huge
because of the high spatial and temporal resolution requirements by the previous
two computational issues. Solving large size systems sometimes is a computational
problem. We call the issue as **Singularity of Membrane Capacitance**.

Fifth, the conductivity tensor $D(x)$ in the cardiac problem (1.1) presents a computa-
tional issue as well. Notice that $D(x)$ depends on the spatial variable $x$. Actually,
it is closely related to the rotational anisotropy and inhomogeneity of electrical con-
ductivities, the microscopic structure as well as the fiber orientation of the cardiac
tissue. Some numerical methods, such as the finite difference method, have diffi-
culty accurately solving problems with such a conductivity tensor $D(x)$. In addition,
the elliptic solver used for the bidomain model, may not be efficiently convergent
to solution if the anisotropy is very strong or the conductivity tensor $D(x)$ changes
too quickly (e.g., when the cardiac tissue is immersed into a bath with different conductivity properties). We call this computational issue as **Inhomogeneity and Anisotropy of Conductivities**.

Finally, notice that the two functions $I_{\text{ion}}(u,q)$ and $M(u,q)$ describe the cardiac membrane dynamics. In effect, the multiscale phenomenon of cardiac excitation is essentially determined by the membrane dynamics (property) $I_{\text{ion}}$ and $M$. In addition, the membrane dynamics may yield another computational issue. Since the functions $I_{\text{ion}}$ and $M$ are generally nonlinear functions, in the case that an implicit time integration scheme is employed to integrate the equations (1.1), nonlinear systems of huge size have to be solved in the solution process. We call this issue as **Nonlinearity of Model Problems**.

The computational issues are summarized in Figure 1.4. At the end of this chapter, we will discuss our strategies (thesis objectives) to tackle down these issues.

### 1.2 Literature Review

In this section, we will present two literature reviews. One is on numerical methods for cardiac problems. Another one is on adaptive methods for general problems.

#### 1.2.1 Numerical Methods for Cardiac Problems

Only a brief review on numerical methods for cardiac problems is addressed in this subsection. For a more detailed literature survey, refer to [55, 181, 182].

The numerical methods previously used in simulation of cardiac electrical activity can be divided into two groups. The methods of the first, larger group are non-adaptive: they use fixed spatial grids and fixed time steps. These methods are implemented both on parallel computers [163, 188] and ordinary workstations [18,
Cardiac Model for Electrical Activity of the Heart:

\[ C_m \frac{\partial u}{\partial t} = \frac{1}{\beta} \nabla \cdot D(x) \nabla u - I_{ion}(u, q) \quad x \in \Omega, \ t > 0, \]

\[ \frac{\partial q}{\partial t} = M(u, q) \quad x \in \Omega, \ t > 0. \]

1. Complexity of Domain: \( \Omega \)

2. Nonlinearity of Model Problems: \( I_{ion}, M \)

3. Spatial Locality of Electrical Wave Fronts: \( x \)

4. Multiscale Phenomena of Cardiac Excitation: \( t \)

5. Inhomogeneity and Anisotropy of Conductivities: \( D \)

6. Singularity of Membrane Capacitance: \( C_m \)

Figure 1.4: Issues from computational electrophysiology
132, 160]; they use both explicit [61, 132, 160, 163] and semi- or fully implicit time-stepping techniques [18, 96, 107, 158, 188, 190]. While clever implementation allowed the researchers to conduct some sample three-dimensional simulations, the resulting modeling tool is very expensive. For example, active anisotropic models that run on parallel machines (256 processor CM5) have been reported to require more than 15 hours to simulate 100 ms of action potential propagation in a $0.8 \times 0.8 \times 0.1$ cm piece of the cardiac muscle [163]. The main reason for this expense is the weakness common to all the methods from this group. When algorithms fail to recognize regions of high electrical activity and to apply separate numerical treatment depending on the level of electrical activity, the resulting codes suffer stringent limits on the size of the time step and spatial resolution for the entire calculation.

The methods of the second group employ spatial adaptivity, temporal adaptivity, or both in order to avoid the limitation above. The intent of these algorithms is to concentrate the computational work in regions of high electrical activity. Because the positive-definite conductivity tensors in the reaction diffusion systems (see Chapter 2) are relatively small compared to the reaction rate, these regions of high electrical activity will have a small width. But, it is noteworthy that, during tachycardia the propagating waves may be very close together, making spatial adaptivity more difficult.

An adaptive scheme could reduce the total computational work by the factor given by the ratio of the domain volume to the total volume of the regions of high electrical activity. This upper bound on the adaptive speedup is problem-dependent. If the reaction rates are large compared to the conductivities, these regions of high electrical activity will be small, and the potential speedup from an adaptive mesh refinement (AMR) scheme is large.

Several authors have used such adaptive algorithms successfully. A domain-
decomposition method combined with an alternating direction implicit (ADI) Rush-Larsen method implemented in [155] dynamically tracks active regions, decomposes the region of computation into small subdomains and uses explicit time stepping in the subdomains (locally) and implicit method for global integration. This algorithm is temporally adaptive, but not spatially adaptive. The authors report time savings on the order of $3 - 17$, compared with a non-adaptive technique. Significantly larger two-dimensional models with active membrane dynamics can therefore be used for simulations. A similar approach, involving a combination of an implicit integration technique with multigrid iteration, is employed in [142]. A large modular code has recently been developed to accommodate a wide variety of existing cardiac models and numerical approaches in [152]. This code is not adaptive in space. It allows the user to choose an adaptive time-integration technique among a few explicit, semi-implicit and implicit methods. The linear algebra is handled by a choice of an iterative method (CG, GMRES) with a preconditioner (SOR, incomplete Cholesky). An irregular grid option allows for complex geometries. Finally, the code is organized to allow parallelization. The lack of spatial adaptivity prevents the direct extension of these methods into three spatial dimensions. Electric stimulation of the tissue by strong shocks especially calls for spatial adaptivity in the regions near the electrodes. The method is less than optimal for such defibrillation studies even in two dimensions.

Rush-Larsen temporal adaptation technique was also considered in [150]. Here, it was combined with explicit, semi-implicit and fully implicit methods, as well as with spatial adaptation technique. The active front was tracked dynamically, all the nodes were marked as either “active” (near the front) or “inactive” and the calculations were performed only at the “active” nodes. This approach does not allow control of the error in the transmembrane potential as it accumulates over time. This model also
seems to be only applicable for propagations studies.

A space adaptive algorithm for the cardiac bidomain model of extracellular potential is developed by Pennacchio [145]. In the algorithm, the computational domain is decomposed into subdomains and each subdomain is independently and uniformly refined. The subdomains belonging to regions of high electrical activity use finer grids than others. The matching of different discretizations on adjacent subdomains is weakly enforced using the mortar finite element method [28, 29].

Methods that are adaptive both temporally and spatially are only beginning to be developed. A space-time adaptive approach that uses finite differences and is explicit in time for the diffusion term has already been shown to produce a factor of 5 reduction in 2D (and 50 reduction in 3D) computational time and memory expense [54, 55, 56], relative to the non-adaptive explicit algorithm. Asymptotically accurate a posteriori error estimates for spatially adaptive finite difference methods for parabolic equations in three-dimensions have been derived and the methods have been shown to converge on irregular grids [130]. Although not yet applied directly to the reaction-diffusion systems (1.1) from computational electrophysiology, the results show promise for the spatially adaptive approach.

Recently, Trangenstein [181, 182] uses a both space and time adaptive mesh refinement method for the FitzHugh-Nagumo and the Luo-Rudy phase one models. In the algorithm, an operator splitting technique is first applied to separate the linear diffusion term from the nonlinear reaction terms. The resulting nonlinear reaction terms are spatially-independent and can be integrated locally and adaptively with a standard adaptive (stiff) ODE integrator. For the linear diffusion, to avoid the time step restriction associated with an explicit scheme, an implicit Crank-Nicolson finite element integration method is employed together with a standard multigrid iteration for the resulting linear systems. The overall time stepping follows the AMR approach.
proposed by Berger and Oliger [27],

In addition, it is noteworthy that most methods employed in computational electrophysiology, up to now, are using the first-order forward or backward Euler method for time integration. Sometimes, higher order methods, such as Runge-Kutta schemes, are used together with a semi-implicit integration [153]. However the semi-implicit treatment of the system essentially restricts the simulation to be only of first-order in time. It seems that, with an operator splitting technique, Qu and Garfinkel [154] and the Oslo group [173] achieve a truly second-order accuracy in both space and time.

1.2.2 Adaptive Methods for General Problems

Since the thesis work focuses on the application of an adaptive mesh refinement (AMR) algorithm to the reaction-diffusion problems arising from modeling the electrical activity of the heart, a short literature review on adaptive methods will be addressed in the subsection. Due to the space and time limitation, we omit the review on a posteriori error estimation, which is also an important part of an adaptive algorithm. For detailed and complete discussion on this topic, refer to the works by J. T. Oden and Ainsworth [137] and R. Verfürth [185, 186].

Adaptive methods for partial differential equations from science and engineering applications have been extensively and widely used since its first introduction in the late 1970s [7, 45]. The applications include elliptic, parabolic, hyperbolic and mixed type problems, from structural mechanics, fluid dynamics, gas dynamics, magneto-hydrodynamics and electrophysiology etc. [5, 10, 15, 24, 26, 31, 55, 67, 68, 79, 97, 124, 126, 165, 179].

Adaptive methods may be classified into different categories in different ways. Based on the different adaptation strategies used, there are three basic adaptive
methods: mesh enrichment (h-refinement) [130, 131], order variation (p-refinement) [9, 174] and moving mesh (r-refinement) [1, 10, 91, 92, 127, 149]. In an h-refinement method, points or elements are added to the regions where the refinement is required or removed from the areas where the refinement is not necessary. In the p-refinement method, the order of local discretization polynomials is increased in the places where the solution is smooth and decreased in the parts where the solution is estimated to be varying sharply or discontinuous [9, 174]. In the r-refinement approach, the mesh size and topology is conserved during the solution process, but the nodes are moved around by an equidistribution principle [7, 98] or harmonic maps [118]. Each of the strategies has its own advantages and disadvantages [10, 119, 177]. Sometimes, they are used in combinations such as the hp-adaptive methods [2, 66, 136].

Based on the different grid structures employed, the adaptive methods are divided into two classes: unstructured grid methods, structured or block-structured grid methods [162]. The latter one includes Cartesian grid methods [3, 25, 116, 117, 144, 148, 159].

Unstructured grids are usually created through front advancing techniques or Delaunay triangulations. It is also common that people use irregular red-green refinement, longest edge bisection, newest vertex bisection etc. to generate unstructured grids [16, 13]. In order to preserve the shape regularity of a grid, special techniques or procedures need to be applied in the grid generation. Unstructured grids are suited for geometrically complex domains and widely used in the communities of finite element method, finite volume method, and discontinuous Galerkin method etc.. Typically, triangles in two space dimensions (2D) and tetrahedrals in three space dimensions (3D) are used to represent unstructured grids.

A structured or block-structured grid is often generated from a global or local regular refinement of a structured or unstructured base grid [52, 162]. The generation
of a structured grid is relatively easier than that of unstructured grid. By regular refinement, including bisection in 2D and trisection in 3D, the newly generated elements inherit shape regularity from their parents automatically. No special technique is required to preserve the shape regularity of a grid. However, hanging nodes may arise during (local) regular refinement. A specific procedure is required such that the solution is continuous or weakly continuous across the hanging nodes [16, 38, 131].

Structured grids are widely used together with finite difference and spectral element methods. Rectangles or hexahedrals are usually employed for these grids.

Based on the different data structures used in the implementation of mesh adaptation [23], there are four main approaches. The first one, which is the oldest, uses linked lists of nodes, edges, sides (3D) and elements to represent grids [12]. It is commonly used for unstructured grids. The length of a list is dynamically changed during mesh adaptation. The insertion or deletion of a new entity influences the global ordering and the length of a list. This presents some problems such as indirect addressing for efficient computer cache usage.

The second one uses cell-based tree structures, which include the quad-trees in 2D and the octrees in 3D [60, 110, 151]. Each node in the tree structure corresponds to an element in the grid represented. The refined elements from a large element are called the children of the corresponding node in a tree. The mesh refinement ratio is limited by the quadtree or octree structures to be two. The node-based tree method has the indirect addressing problem as well.

The third one employs block-based tree structures [51, 171]. Different from the cell-based tree approach, each tree node in the block-based tree structure is a block, which consists of a logically rectangular array of elements. This approach solves the indirect addressing problem to some extent. That the refinement ratio can be chosen to be any integer makes the method more flexible for some typical multiscale
problems. In this approach, each block is associated with a number of ghost cells. In higher dimensions, smaller the refinement ratio, larger is the ratio of the number of ghost cells to interior cells, which means a more severe waste of computational resources. Typically people try to keep the ratio of the number of ghost cells to the number of interior cells to be small by choosing not to use many small grid “blocks”. That is, each block consists of more elements. However, this may involve too many unnecessarily refined elements and hence too much additional computational overhead.

The fourth approach, called adaptive mesh refinement (AMR) and originally proposed by M. Berger and J. Oliger in 1980s [24, 27], uses clustered logically rectangular blocks or patches to represent computational grids. It solves the indirect addressing problem and achieves an almost minimal number of unnecessarily refined elements. Together with a special time stepping algorithm, this approach is well suited for the time-dependent problems that involve propagation of sharp fronts or discontinuities. However, the cell clustering algorithm employed in this approach works easily only for Cartesian grids or those which can be defined by a single index space. It is not straightforward for this approach to deal with front propagations on complex domains. In addition, Berger-Oliger’s approach may have load balancing problem for distributed computing if the number of grid blocks is not at least three times the number of processors. Other AMR approaches involve more distributed communication than Berger’s and are harder to guarantee that grid neighbors are on same processor or all neighbors of cells on a given processor are on small number of other processors.

There are variations or combinations of the strategies above. For example, in order to deal with problems on complex geometries, P. Colella etc. combines a boundary embedded method with Berger-Oliger’s approach [102, 103]. D. Quinlan etc. uses
some composite grid techniques to deal with the moving boundary problems [114]. However, for these variations, it is difficult to achieve higher order, greater than two, accuracy for problems on complex domains.

Remark 1.1: The AMR algorithm proposed in the thesis work is aimed to be geometrically flexible as much as unstructured grids based strategies, as easy as structured grids based methods to be implemented, and as efficient as Berger-Oliger’s approach in time stepping. The AMR algorithm is designed and implemented in a unified dimension independent way. A unified code enables us debug the program in a lower dimension and then naturally extend to higher dimensions.

The AMR algorithm falls into the category of the mesh refinement (h-refinement) methods. It uses lists of rectangular geometric entities to represent structured grids. It is well suited for wave propagation problems on complex geometries. By using higher order finite elements in the AMR, an arbitrary order accuracy can in principle be obtained even if the underlying computational domain is as complex as the heart. Hopefully, the AMR algorithm may perform well in distributed computing using space filling curves to partition a grid into load balanced sub-grids while keeping the communication overhead minimized.

1.3 Thesis Objectives

As discussed, we must take into account realistic fiber geometry, rotational anisotropy of cardiac conductivities, detailed membrane properties, microscopic structure of the heart tissue and the inhomogeneous nature of myocardium in numerically modeling the electrical activity of the heart. To solve the six computational issues discussed in subsection 1.1.2 (see Figure 1.4), we propose five strategies as follows.
1. **Operator Splitting Techniques**

Operator splitting techniques enable us to decouple the nonlinear reactions, which describe the membrane dynamics $I_{\text{ion}}(V, q)$ and $\mathcal{M}(V, q)$ in (1.1), from the linear diffusion. As a result, the nonlinear reactions are spatially independent, which implies that the size of nonlinear equations is greatly reduced and so are the computational difficulty and cost. On the other hand, the resulting diffusion is linear even though it is spatially dependent. With a space discretization method, the linear diffusion can be easily integrated. In this way, the issue of *nonlinearity of model problems* is settled down.

2. **Stiffly Accurate A-stable Schemes**

A second-order singly-diagonally implicit Runge-Kutta (SDIRK) scheme will be employed to integrate the stiff nonlinear reactions resulting from the application of operator splitting to the cardiac problems. The SDIRK scheme is absolutely stable (A-stable) on the left half complex plane and is stiffly accurate. With the stiffly accurate A-stable SDIRK scheme, the computational issue related to the multiscale phenomena of cardiac excitation is overcome.

3. **Finite Element Discretization**

It is well-known that finite element methods are well suited for problems on geometrically complex domains. By using isoparametric quadrilateral (2D) or hexahedral (3D) elements, we can accurately approximate the boundaries of the complex computational domain and boundary conditions. With finite element methods for the linear diffusion, the issue of the *inhomogeneity of conductivity*, related to the rotational fiber orientation and the microstructure of the cardiac tissue etc., is naturally solved as long as the conductivity tensor $\mathbf{D}(\mathbf{x})$ is smooth enough.
4. **Adaptive Mesh Refinement Algorithm**

It is exactly the spatial locality of electrical wave fronts that makes an adaptive mesh refinement strategy well suited for numerically modeling the electrical activity of the heart. By locally refining the regions in a grid where the electrical wave fronts are present or the solution data have large errors, an adaptive mesh refinement algorithm can greatly reduce the size of grids and hence the number of unknowns. A both space and time adaptive strategy will further improve the simulation efficiency.

*In the thesis work, we will propose an AMR algorithm, which aims to be geometrically flexible as much as unstructured grids, as easy as structured grids to be implemented, and as efficient as Berger-Oliger’s approach in time stepping.*

5. **Multilevel/Multigrid Iteration**

The singularity of membrane capacitance in the cardiac model (bidomain model) determines that an implicit (elliptic) solver must be applied to solving the degenerate linear diffusion equations. In the thesis work, we will use the Crank-Nicolson scheme to discretize the equations in time. The implicit integration results in linear systems. To efficiently solve the linear systems, a multigrid/multilevel iteration on locally refined grids is scheduled to be employed. With the optimal multigrid iteration, the singularity of membrane capacitance is no longer a computational issue.

*It is noteworthy that a multilevel/multigrid iteration may have a convergence problem with very strong anisotropic conductivities. Fortunately, the anisotropy ratio of experimentally measured effective conductivities in the cardiac tissue varies in the range from 5.7 to 10.8 [59, 133, 157]. The size of the anisotropy ratio is not too large. Hopefully the multigrid iteration employed in the thesis*
work will still perform very well.

In the thesis work, we also aim to propose a novel multigrid smoothing (relaxation) procedure for composite grid equations on locally refined grids.

In short, in the thesis work, a second-order operator splitting technique is first employed to decouple the (degenerate) linear diffusion from the stiff nonlinear reactions in a cardiac model like (1.1). Then the resulting stiff nonlinear reactions are locally integrated with a second-order SDIRK scheme. The linear diffusion is discretized by a second-order conforming finite element method in space and the second-order Crank-Nicolson scheme in time. During the solution process, the grids are locally, dynamically and automatically generated by the proposed adaptive mesh refinement (AMR) algorithm. The linear systems on the adaptively refined grids are solved by the standard $V$-cycle multigrid algorithm.

### 1.4 Outline of Thesis

The model problems for the thesis work are presented in Chapter 2. First, we describe the microscopic bidomain model based on the assumption that the cardiac tissue is idealized as a periodic structure of unit cells. Then, the macroscopic bidomain model is derived from the microscopic model by a homogenization process. Next, the simplified monodomain model is described. We also discuss some systems for the membrane current dynamics, which must be supplied to make the bidomain or monodomain model complete. It is well-known that each of the bidomain and monodomain models is governed by a singularly perturbed reaction-diffusion system coupled with a set of stiff nonlinear ordinary differential equations. There are traveling wave phenomena associated with these kinds of problems. In general, it is very difficult to make theoretical analysis for the reaction-diffusion systems. This
motivates us to perform numerical simulations.

In Chapter 3, the numerical methods used to solve the singularly perturbed reaction-diffusion systems are described in detail. Due to the stability restriction on time step size associated with explicit schemes, it is better to treat both reaction and diffusion terms implicitly to solve the cardiac model problems. However, a fully implicit integration requires us to solve a large nonlinear system in each time step. As a solution to the problem, operator splitting techniques are employed to decouple the reaction and diffusion terms in the reaction-diffusion system to be solved. After that, both terms can be integrated implicitly and independently. For the stiff nonlinear ordinary differential equations, a second-order singly-diagonally implicit Runge-Kutta scheme is employed. In order to achieve a fully adaptive time integration, some techniques for error estimation and step size control are addressed. Next, a piece-wise linear finite element discretization for the linear diffusion resulting from operator splitting is described. The discretization together with the Crank-Nicolson temporal approximation has second-order accuracy. Since we are working with quadrilateral and hexahedral elements in the AMR algorithm, hanging nodes appear on coarse-fine grid interfaces. A technique to deal with the hanging nodes is also presented. As a result, the finite element discretization is conformal.

The adaptive mesh refinement algorithm is the central part of the thesis work. In Chapter 4, the details of the algorithm are discussed. First, we make a few basic assumptions for the AMR algorithm, similar to the approach proposed by Trangenstein. After that, a typical grid representation strategy is described, using lists of grid entities such as cells, sides, edges, and nodes. The grid representation is complete in the sense that the connectivities among grid entities are well established. Next, a special ordering of the grid entities is proposed, which makes the coarse-fine grid interface operations convenient and easy to implement. The time stepping strategy of
the AMR algorithm follows Berger-Oliger’s approach. The typical modules, including recursive integration, time step selection, tagging of cells, mesh regridding, data up-/downscaling, are briefly described in the middle of the chapter. As an essential part of an adaptive algorithm, a Richardson extrapolation based error estimation procedure is prescribed in the last section.

The standard V-cycle multilevel/multigrid algorithm for the composite grid equations resulting from the conforming finite element discretization of linear diffusion on adaptively refined grids is discussed in Chapter 5. First, we present the abstract theory for subspace correction methods, which include the standard multigrid methods and other domain decomposition methods. Next, the basic components of the V-cycle multigrid algorithm: relaxation/smoothing and coarse grid correction, are described for both that on uniformly refined grids and that on adaptively refined grids. Some implementation details such as the initial guess, the stopping criterion and the coarsest grid solver for the multigrid iteration are addressed finally.

Numerical experiments with the AMR algorithm are presented in Chapter 6. First, we describe the tolerances, parameters and modules used in the experiments. Next, we demonstrate that the convergence rate of our AMR algorithm is second-order by numerical results for a few singularly perturbed reaction-diffusion problems, whose exact solutions are known. After that, we present and analyze computational times for the FitzHugh-Nagumo model problems in both 1D and 2D to show the numerical accuracy and algorithm efficiency of the AMR algorithm. The geometry flexibility of the algorithm is also demonstrated. At the end, the application of the AMR algorithm developed in the thesis work to more realistic models is discussed and numerical results for the Luo-Rudy phase one bidomain model are presented.

In the final chapter, we conclude the thesis with a few conclusions and the main achievements are summarized. We also point out some possible future work and
improvement of the AMR algorithm to the cardiac model problems. They include efficient parallelization of the algorithm, conformal refinement with triangular and tetrahedral elements, incorporation of higher order methods, and even a fully implicit discretization with nonlinear multigrid solver.
Chapter 2

Model Problems

Model problems for the thesis work are presented in this chapter. First, a homogenization process based on asymptotic analysis for periodic structures of the heart tissue is described in section 2.1, which results in the macroscopic bidomain model. Next, the simplified monodomain model is derived in section 2.2, under the assumption of equal anisotropic ratios. A few cardiac models of membrane dynamics are discussed in section 2.3. Finally, it is emphasized that the cardiac models are singularly perturbed reaction-diffusion systems, which involve traveling wave phenomena.

2.1 The Bidomain Model

The discussion in the section is adapted mainly from Neu and Krassowska [133] and partly from Colli-Franzone and Savare [76]. For a formal derivation of the bidomain model and modeling details, see the original papers, the works by Keener and Sneyd [108] and the references therein.

The bidomain model [81, 184] arising from computational cardiology takes into account realistic fiber geometry, anisotropy of cardiac conductivities, detailed membrane properties, microscopic tissue structure, and the inhomogeneous nature of myocardium [134, 160, 161, 183]. With a proper choice of parameters, it is described by a system of singularly perturbed reaction-diffusion partial differential equations coupled with a set of stiff ordinary differential equations representing cell membrane dynamics. Specifically, the bidomain model consists of the equations for the intra- and extracellular potentials, $\Phi_i$ and $\Phi_e$, coupled through the transmembrane poten-
potential, \( V_m = \Phi_i - \Phi_e \), for \( x \in \Omega \) and \( t > 0 \):

\[
\nabla \cdot D_i \nabla \Phi_i = \beta \left\{ C_m \frac{\partial V_m}{\partial t} + I_{\text{ion}}(V_m, q) \right\},
\tag{2.1a}
\]

\[
\nabla \cdot D_e \nabla \Phi_e = -\beta \left\{ C_m \frac{\partial V_m}{\partial t} + I_{\text{ion}}(V_m, q) \right\},
\tag{2.1b}
\]

\[
\frac{\partial q}{\partial t} = M(V_m, q),
\tag{2.1c}
\]

where \( \Omega \subset \mathbb{R}^d \) is the bounded physical domain occupied by the cardiac tissue; \( q \) is a set of state variables such as ionic concentrations which define the physiological state of the cellular structures; \( \beta \) is a surface-to-volume ratio of cardiac cells, \( C_m \) is the membrane capacitance per unit area and \( D_i, D_e \) are specific conductivity tensors in intra- and extracellular spaces respectively; \( I_{\text{ion}}(V_m, q) \) and \( M(V_m, q) \) are functions approximating the cellular membrane dynamics \([94, 133, 153]\). For the units of all quantities mentioned above, see the section next.

**Remark 2.1:** It is noteworthy that, if the equations (2.1a) and (2.1b) are written out in terms of \( \Phi_i \) and \( \Phi_e \) using the fact that \( V_m = \Phi_i - \Phi_e \), the coefficient of the first derivative of time is a symmetric and positive semidefinite matrix. We denote the coefficient matrix by

\[
C_m = \begin{pmatrix} C_m & -C_m \\ -C_m & C_m \end{pmatrix}
\tag{2.2}
\]

and call it the membrane capacitance matrix. Due to the singularity of the capacitance matrix \( C_m \), the equations (2.1a)-(2.1b) actually make up a degenerate parabolic reaction-diffusion system.

**Remark 2.2:** In the literature, the partial differential equations (2.1a) and (2.1b) in the bidomain model are sometimes formulated as a system of elliptic-parabolic
equations \[146, 173, 189\]:

\[
\nabla \cdot (D_i + D_e) \nabla \Phi_e = -\nabla \cdot D_i \nabla V_m, \\
(2.3a)
\]

\[
-\beta \left\{ C_m \frac{\partial V_m}{\partial t} + I_{ion}(V_m, q) \right\} = \nabla \cdot D_e \nabla \Phi_e. \\
(2.3b)
\]

Here, the first equation (2.3a) is an elliptic equation and the second one (2.3b) is a parabolic equation.

In the thesis work, we assume that the cardiac tissue is insulated. Hence, homogeneous Neumann boundary conditions of the following form

\[
\mathbf{n}^T \cdot D_i \nabla \Phi_i = 0 \quad \text{and} \quad \mathbf{n}^T \cdot D_e \nabla \Phi_e = 0,
(2.4)
\]

where \( \mathbf{n} \) is the unit outward normal to the boundary \( \partial \Omega \), are imposed on \( \partial \Omega \) all the time. The boundary conditions (2.4) together with appropriate initial conditions at \( t = 0 \), e.g.,

\[
\Phi_i(x, 0) = \Phi_{i,0}(x), \quad \Phi_e(x, 0) = \Phi_{e,0}(x) \quad \text{and} \quad q(x, 0) = q_0(x), \quad (2.5)
\]

make the bidomain equations (2.1) solvable. Note that the intra- and extracellular potentials \( \Phi_i \) and \( \Phi_e \) are unique up to a constant. We may choose a reference potential to fix them by: (a) constraining the value of \( \Phi_i \) or \( \Phi_e \) at a point in the domain \( \Omega \) to be zero; (b) or, alternatively, translating \( \Phi_i \) or \( \Phi_e \) such that its average over the domain \( \Omega \) is zero.

As a macroscopic model, the bidomain model can be derived from a microscopic model by a homogenization process, which averages intra- and extracellular electric potentials and currents. In the following, we will discuss the microscopic cellular model first.
2.1.1 Introduction

At a microscopic level, the cardiac cellular structure of the tissue can be viewed as composed of two volumes: the intra-cellular space (inside the cells) and the extra-cellular space (outside the cells) separated by the active membrane [76]. More specifically, the cardiac tissue consists of elongated cells connected together by low resistance end-to-end and/or side-to-side junctions and surrounded by an extracellular fluid.

For the derivation of the bidomain model, the microstructure of such a cellular medium is idealized: all cells have the same shape and are arranged in space in a regular fashion, and have the same pattern of connections with neighbors. Thus, the real structure of the tissue is approximated by a spatially periodic model [133]. The smallest repeatable pattern of this structure is called the unit cell.

There are a few fundamental material constants associated with the cellular media:

$d_c$: A typical measure of the cell dimension (units: $L$) (e.g. the cell diameter of a unit cell).

$\sigma_i$: Conductivity of the cytoplasm filling the inside of cell (units: $R^{-1}L^{-1}$). The conductivity of the fluid filling the extracellular space $\sigma_e$ has the same order of magnitude as $\sigma_i$ and, hence, does not need to be considered a fundamental material constant.

$R_m$: Surface resistivity of the membrane that separates the inside and the outside of cell ($RL^2$).

$C_m$: Capacitance of the membrane ($R^{-1}L^{-2}T$).

In the definitions above, $R$ denotes an arbitrary unit of resistivity, $L$ is a unit of length, and $T$ a unit of time.

Let $\Omega_i$, $\Omega_e$ be respectively the intra- and extracellular domains, which are sepa-
rated by the membrane $\Gamma$; see Figure 2.1. At the microscopic level, the intra- and extracellular potentials $\Phi_i$ and $\Phi_e$ are solutions to Laplace equations:

$$\Delta \Phi_i = 0 \quad \text{in } \Omega_i, \quad (2.6a)$$
$$\Delta \Phi_e = 0 \quad \text{in } \Omega_e. \quad (2.6b)$$

In the case that a stimulation current is applied to the intra- and extracellular spaces, the potentials are instead governed by Poisson equations:

$$\Delta \Phi_i = \rho_i^s \quad \text{in } \Omega_i, \quad (2.7a)$$
$$\Delta \Phi_e = \rho_e^s \quad \text{in } \Omega_e. \quad (2.7b)$$

Here, $\rho_i^s$ and $\rho_e^s$ denote the current densities due to the stimulation applied. For the sake of simplicity, in the rest of this section we only consider the case that no current stimulation is applied.

Boundary conditions that are applied to $\Phi_i$ and $\Phi_e$ on $\Gamma$ embody the biophysical properties of the membrane. They are given by

$$-\sigma_i \partial_n \Phi_i = C_m \frac{\partial}{\partial t} V_m + I_{ion}(V_m, q), \quad (2.8a)$$
$$-\sigma_e \partial_n \Phi_e = C_m \frac{\partial}{\partial t} V_m + I_{ion}(V_m, q), \quad (2.8b)$$
$$\frac{\partial q}{\partial t} = M(V_m, q), \quad (2.8c)$$

on the membrane $\Gamma$, where the normal derivative $\partial_n$ is taken with the normal pointing outside of the cell and the transmembrane potential

$$V_m = \Phi_i - \Phi_e \quad \text{on } \Gamma \quad (2.9)$$
is the difference between $\Phi_i$ and $\Phi_e$; other parameters or variables are the same as in
the bidomain model (2.1). The left hand sides in (2.8) represent the intra- and extra-
cellular currents evaluated on $\Gamma$. The right hand sides in (2.8) describe the current
across the membrane as having two components. The capacitive component depends
on the membrane capacitance $C_m$ and the time derivative of the transmembrane
potential $V_m$. The nonlinear current-voltage relationship $I_{\text{ion}}(V_m, q)$ and $M(V_m, q)$
corresponds to the chosen excitability model of the membrane dynamics.

The model described by the equations (2.6) and (2.8) gives a microscopic repre-
sentation of the electric potentials in the heart. In order to go further to derive the
macroscopic bidomain model, we need to look back to the four fundamental material
constants $d_c, \sigma_i, R_m$ and $C_m$.

First, a dimensionless combination of these parameters may be formed:

$$
\epsilon = \left( \frac{d_c}{R_m \sigma_i} \right)^{1/2}.
$$

(2.10)

For values typical of cardiac muscle, $\epsilon$ is small, e.g., $\epsilon = 4 \cdot 10^{-3}$. The dimensionless
parameter $\epsilon$ is closely associated with the characteristic scale of the microscopic
model, as we will see.

Using the four fundamental material constants $d_c, \sigma_i, R_m$ and $C_m$, several time
and length constants can be formulated too. For convenience, the macroscopic units
of length as $L$ is defined as $L = d_c/\epsilon$ and the time constant associated with charging
the membrane by the transmembrane current is given by $T = R_m C_m$. After that, we
can convert the cellular problem into a non-dimensional form, by scaling space and
time with the constants, i.e.,

$$
\hat{x} = x/L, \quad \hat{t} = t/T,
$$

(2.11)

and scaling the potentials $\Phi_i$, $\Phi_e$ and $V_m$ by a convenient unit $\Delta V$ of measure for
them, i.e.,

\[ \hat{\Phi}_i = \Phi_i / \Delta V, \quad \hat{\Phi}_e = \Phi_e / \Delta V, \quad \hat{V}_m = V_m / \Delta V. \]  

(2.12)

Assume that the typical state variables \( q \) are dimensionless, or they are scaled by \( \hat{q} = Q^{-1}q \) with \( Q \) a convenient unit (matrix) of measure for the state variables. The non-dimensional current-voltage relationship \( I_{ion}(V_m, q) \) has the following form

\[ \hat{I}_{ion}(\hat{V}_m, \hat{q}) = \frac{R_m}{\Delta V} I_{ion}(V_m, q), \]

and the dimensionless nonlinear reactions for state variables

\[ \hat{M}(\hat{V}_m, \hat{q}) = \frac{T}{Q} M(V_m, q). \]

Furthermore, after simple calculation for the equations (2.8), we obtain the following dimensionless boundary conditions:

\[ -\partial_{n}\Phi_i = \epsilon \left\{ C_m \frac{\partial}{\partial t} V_m + I_{ion}(V_m, q) \right\}, \]  

(2.13a)

\[ -\mu \partial_{n}\Phi_e = \epsilon \left\{ C_m \frac{\partial}{\partial t} V_m + I_{ion}(V_m, q) \right\}, \]  

(2.13b)

\[ \frac{\partial q}{\partial t} = M(V_m, q), \]  

(2.13c)

where \( \mu \) is the ratio of extracellular and intracellular conductivities

\[ \mu = \frac{\sigma_e}{\sigma_i} \]  

(2.14)

and \( \epsilon \) is the dimensionless small parameter defined in (2.10). For convenience, the superscripts \(^*\) of the dimensionless variables are omitted.

The Laplace equations in (2.6) are invariant with respect to the scaling above.
2.1.2 Homogenization

As an asymptotic analysis technique, the homogenization process is typically used for engineering problems involving multiscale phenomenon. In particular, for boundary value problems with periodic structure, the period of the structure is assumed to be very small compared to the size of the domain of interest. Then perturbation analysis is employed to obtain an asymptotic expansion of the solution in terms of a small parameter, which is the ratio of the period of the structure to a typical length in the region [22].

Before presenting the two-scale asymptotic expansion, let us first introduce a few symbols.

As stated, the cardiac tissue is idealized as a periodic structure of unit cells which are connected by end-to-end and side-by-side junctions. The periodic structure is similar to a regular lattice of interconnected cylinders. Let $Y_i$ and $Y_e$ denotes portions of intra- and extracellular spaces $\Omega_i$ and $\Omega_e$ that belong to a unit cell, respectively. Similarly, the unit cell portion of the membrane $\Gamma$ is denoted by $S$; see Figure (2.2) for an illustration.

Introducing a microscopic variable $\xi = x/\epsilon$, associated with the dimension of a unit cell, we assume the electric potentials $\Phi_i$, $\Phi_e$, $V_m$ and the state variables $q$ are functions of both the slow macroscopic variable $x$ and the fast microscopic variable $\xi$. The potentials and state variables have the following asymptotic expansion in powers
of the dimensionless parameter $\epsilon$:

$$\Phi_i(x, \xi) = \Phi_i^{(0)}(x, \xi) + \epsilon \Phi_i^{(1)}(x, \xi) + \epsilon^2 \Phi_i^{(2)}(x, \xi) + \cdots,$$  \hspace{1cm} (2.15a)

$$\Phi_e(x, \xi) = \Phi_e^{(0)}(x, \xi) + \epsilon \Phi_e^{(1)}(x, \xi) + \epsilon^2 \Phi_e^{(2)}(x, \xi) + \cdots,$$  \hspace{1cm} (2.15b)

$$V_m(x, \xi) = V_m^{(0)}(x, \xi) + \epsilon V_m^{(1)}(x, \xi) + \epsilon^2 V_m^{(2)}(x, \xi) + \cdots,$$  \hspace{1cm} (2.15c)

$$q(x, \xi) = q^{(0)}(x, \xi) + \epsilon q^{(1)}(x, \xi) + \epsilon^2 q^{(2)}(x, \xi) + \cdots.$$  \hspace{1cm} (2.15d)

The slow and fast variables correspond respectively to the global and local structure of the field. The coefficients are $1-$periodic functions of $\xi$.

Let $\nabla_x$ and $\nabla_\xi$ denote gradients with respect to $x$ and $\xi$, and the full gradient operator

$$\nabla = \frac{1}{\epsilon} \nabla_\xi + \nabla_x.$$  \hspace{1cm} (2.16)

The full Laplacian operator $\Delta$ is represented as

$$\Delta = \frac{1}{\epsilon^2} \Delta_{\xi\xi} + \frac{1}{\epsilon} (\nabla_\xi \cdot \nabla_x + \nabla_x \cdot \nabla_\xi) + \Delta_{xx}.$$  \hspace{1cm} (2.17)

Substituting the asymptotic expansions (2.15a) and (2.15c)-(2.15d) into the Laplace equation (2.6a) and equating the coefficients of the powers $-2, -1, 0$ of the dimensionless parameter $\epsilon$ to zero, we obtain the following equations:

$$\frac{1}{\epsilon^2} : \quad \Delta_{\xi\xi} \Phi_i^{(0)} = 0 \hspace{1cm} \text{in} \ Y_i,$$  \hspace{1cm} (2.18a)

$$\frac{1}{\epsilon} : \quad \Delta_{\xi\xi} \Phi_i^{(1)} + (\nabla_\xi \cdot \nabla_x + \nabla_x \cdot \nabla_\xi) \Phi_i^{(0)} = 0 \hspace{1cm} \text{in} \ Y_i,$$  \hspace{1cm} (2.18b)

$$1 : \quad \Delta_{\xi\xi} \Phi_i^{(2)} + (\nabla_\xi \cdot \nabla_x + \nabla_x \cdot \nabla_\xi) \Phi_i^{(1)} + \Delta_{xx} \Phi_i^{(0)} = 0 \hspace{1cm} \text{in} \ Y_i.$$  \hspace{1cm} (2.18c)

Similarly, substituting the asymptotic expansions into the boundary condition
equation (2.13a), we obtain the following equations:

\[ \frac{1}{\epsilon} : \mathbf{n} \cdot \nabla_\xi \Phi_i^{(0)} = 0 \quad \text{on } S, \quad (2.19a) \]
\[ 1 : \mathbf{n} \cdot (\nabla_\xi \Phi_i^{(1)} + \nabla_x \Phi_i^{(0)}) = 0 \quad \text{on } S, \quad (2.19b) \]
\[ \epsilon : \mathbf{n} \cdot (\nabla_\xi \Phi_i^{(2)} + \nabla_x \Phi_i^{(1)}) = \left\{ \frac{1}{\partial t} V_m^{(0)} + I_{\text{ion}}(V_m^{(0)}, q^{(0)}) \right\} \quad \text{on } S. \quad (2.19c) \]

We may treat the macroscopic variable \( \mathbf{x} \) as a parameter in the equations above and three boundary value problems can be formulated as follows.

For the coefficient \( \Phi_i^{(0)} \) of 1 in the asymptotic expansion (2.15a) for the intracellular potential \( \Phi_i \), the first boundary value problem is given by

\[ \Delta_\xi \Phi_i^{(0)} = 0 \quad \text{in } Y_i, \quad (2.20a) \]
\[ \mathbf{n} \cdot \nabla_\xi \Phi_i^{(0)} = 0 \quad \text{on } S. \quad (2.20b) \]

For the coefficient \( \Phi_i^{(1)} \) of \( \epsilon \) in the asymptotic expansion (2.15a) for the intracellular potential \( \Phi_i \), the second boundary value problem is given by

\[ \Delta_\xi \Phi_i^{(1)} \rightleftharpoons (\nabla_\xi \cdot \nabla_x + \nabla_x \cdot \nabla_\xi) \Phi_i^{(0)} = 0 \quad \text{in } Y_i, \quad (2.21a) \]
\[ \mathbf{n} \cdot \nabla_\xi \Phi_i^{(1)} + \mathbf{n} \cdot \nabla_x \Phi_i^{(0)} = 0 \quad \text{on } S. \quad (2.21b) \]

For the coefficient \( \Phi_i^{(2)} \) of \( \epsilon^2 \) in the asymptotic expansion (2.15a) for the intracellular potential \( \Phi_i \), the third boundary value problem is given by

\[ \Delta_\xi \Phi_i^{(2)} \rightleftharpoons (\nabla_\xi \cdot \nabla_x + \nabla_x \cdot \nabla_\xi) \Phi_i^{(1)} + \Delta_{xx} \Phi_i^{(0)} = 0 \quad \text{in } Y_i, \quad (2.22a) \]
\[ \mathbf{n} \cdot \nabla_\xi \Phi_i^{(2)} + \mathbf{n} \cdot \nabla_x \Phi_i^{(1)} \rightleftharpoons \left\{ \frac{1}{\partial t} V_m^{(0)} + I_{\text{ion}}(V_m^{(0)}, q^{(0)}) \right\} = 0 \quad \text{on } S. \quad (2.22b) \]

Then, the coefficients \( \Phi_i^{(0)}, \Phi_i^{(1)} \) and \( \Phi_i^{(2)} \) in the expansion (2.15a) for the intracellular potential \( \Phi_i \) can be computed as functions of \( \xi \) by solving one by one the boundary value problems (2.20)-(2.22) in the local portion \( Y_i \) of a unit cell.
Note that the problems (2.20)-(2.22) above for the expansion coefficients are actually Neumann boundary value problems (BVP). Under reasonable assumptions on the domain \(Y_i\), its boundary \(S\) and the data on the right hand side of problems, the Neumann BVP has a unique solution up to a constant [76]. The result is standard and well-known [139].

From the first BVP (2.20), which only has a constant solution with respect to \(\xi\), we find that the 1-periodic solution \(\Phi_i^{(0)}\) depends only on the slow macroscopic variable \(x\). Actually, it represents a potential average over \(Y_i\) by letting \(\epsilon\) go to zero.

With the \(\xi\) independence of \(\Phi_i^{(0)}\), the second BVP (2.21) becomes:

\[
\begin{align*}
\Delta_{\xi\xi}\Phi_i^{(1)} &= 0 \quad \text{in } Y_i, \quad (2.23a) \\
\mathbf{n} \cdot \nabla_{\xi}\Phi_i^{(1)} + \mathbf{n} \cdot \nabla_x\Phi_i^{(0)} &= 0 \quad \text{on } S. \quad (2.23b)
\end{align*}
\]

It is not difficult to find that the solution of the BVP (2.23) can be represented as:

\[
\Phi_i^{(1)} = -w_i(\xi) \cdot \nabla_x\Phi_i^{(0)} + \tilde{\Phi}_i^{(1)}(x, t),
\]

where \(w_i = (w_i^1(\xi), w_i^2(\xi), w_i^3(\xi))^T\) and its components \(w_i^k(\xi)\) \((k = 1, 2, 3)\), satisfy

\[
\begin{align*}
\Delta_{\xi\xi}w_i^k(\xi) &= 0, \quad \text{in } Y_i, \quad (2.25a) \\
\nabla_{\xi}w_i^k(\xi) \cdot \mathbf{n}_i &= n_{i, \xi_k} \quad \text{on } S, \quad (2.25b)
\end{align*}
\]

where \(\mathbf{n}_i = (n_i, \xi_1, n_i, \xi_2, n_i, \xi_3)^T\) is the unit outward normal to the membrane portion \(S\) with respect to the intracellular portion \(Y_i\). The problems for \(w_i^k(\xi)\) \((k = 1, 2, 3)\) are solvable up to a constant, and the solutions can be fixed, e.g., by the condition \(\int_S w_i^k(\xi) \, d\xi = 0\).

To use the Gauss divergence theorem, the third BVP (2.22) can be rewritten as

\[
\begin{align*}
\nabla_{\xi} \cdot (\nabla_{\xi}\Phi_i^{(2)} + \nabla_x\Phi_i^{(1)}) &= -\Delta_{xx}\Phi_i^{(0)} - \nabla_x \cdot \nabla_{\xi}\Phi_i^{(1)} \quad \text{in } Y_i, \quad (2.26a) \\
\mathbf{n} \cdot (\nabla_{\xi}\Phi_i^{(2)} + \nabla_x\Phi_i^{(1)}) &= -\left\{ \frac{\partial}{\partial t} V_m^{(0)} + I_{ion}(V^{(0)}, q^{(0)}) \right\} \quad \text{on } S. \quad (2.26b)
\end{align*}
\]
Integrating the equation (2.26a) over the unit cell portion $Y_i$ of the intracellular space $\Omega_i$, by the divergence theorem together with the boundary conditions (2.26b), we obtain the following identity:

$$
\int_{Y_i} \left( \Delta_{xx} \Phi_i^{(0)} + \nabla_x \cdot \nabla_\xi \Phi_i^{(1)} \right) d\xi = \int_S \left\{ \frac{\partial}{\partial t} V_m^{(0)} + I_{\text{ion}}(V_m^{(0)}, q^{(0)}) \right\} ds.
$$

(2.27)

Note that the macroscopic component $\Phi_i^{(1)}(x, t)$ in the representation (2.24) of the coefficient $\Phi_i^{(1)}$ is independent of the microscopic variable $\xi$; and the coefficient $\Phi_i^{(0)}$ is independent of $\xi$ as well. Substituting (2.24) into the identity (2.27), we arrive at

$$
\nabla_x \cdot \left\{ \frac{1}{|Y_i|} \left| Y_i \right| I - \int_{Y_i} \nabla_\xi w_i \, d\xi \right\} \cdot \nabla_x \Phi_i^{(0)} = \hat{\beta} \left\{ \frac{\partial}{\partial t} V_m^{(0)} + I_{\text{ion}}(V_m^{(0)}, q^{(0)}) \right\},
$$

(2.28)

where $\nabla_\xi w_i = [\nabla_\xi w_i^1, \nabla_\xi w_i^2, \nabla_\xi w_i^3]$ and $|Y_i|, |S|$ denote the volume and the area of the unit cell portions $Y_i, S$ of the intracellular space $\Omega_i$ and the membrane $\Gamma$ respectively.

Let $\hat{\beta}$ be the dimensionless surface-to-volume ratio, defined by $\hat{\beta} = \frac{|S|}{|Y|}$, where $|Y|$ is the volume of a unit cell. Then we have the following expression from equation (2.28):

$$
\nabla_x \cdot \left\{ \frac{1}{|Y|} \left| Y_i \right| I - \int_{Y_i} \nabla_\xi w_i \, d\xi \right\} \cdot \nabla_x \Phi_i^{(0)} = \hat{\beta} \left\{ \frac{\partial}{\partial t} V_m^{(0)} + I_{\text{ion}}(V_m^{(0)}, q^{(0)}) \right\}.
$$

(2.29)

Define a tensor $\hat{D}_i$ for the integral above by

$$
\hat{D}_i = \frac{1}{|Y|} \left\{ \left| Y_i \right| I - \int_{Y_i} \nabla_\xi w_i \, d\xi \right\}.
$$

(2.30)

Applying the Green formula and taking into account the periodicity of $w_i(\xi)$, we can rewrite the macroscopic tensor (2.30) of intra-cellular conductivity [76, 133] as:

$$
\hat{D}_i = \frac{1}{|Y|} \left\{ \left| Y_i \right| I - \int_S n_i \otimes w_i \, ds \right\}.
$$
Here, \( \mathbf{n}_i \) is the unit outward normal to \( S \) with respect to the intracellular portion \( Y_i \); the symbol “\( \otimes \)” represents a tensor product of vectors.

Finally, we obtain the following equation for the intracellular potential directly from the identity (2.29):

\[
\nabla_x \cdot (\mathbf{\hat{D}}_i \nabla_x \Phi_i^{(0)}) = \hat{\beta} \left\{ \frac{\partial}{\partial t} V_m^{(0)} + I_{\text{ion}}(V_m^{(0)}, q^{(0)}) \right\}.
\]

(2.31)

It is a dimensionless averaged equation. The intracellular potential \( \Phi_i \) is independent of the microscopic scale \( \xi \).

Similarly, we can obtain the dimensionless averaged equation for the extracellular potential:

\[
\nabla_x \cdot (\mathbf{\hat{D}}_e \nabla_x \Phi_e^{(0)}) = -\hat{\beta} \left\{ \frac{\partial}{\partial t} V_m^{(0)} + I_{\text{ion}}(V_m^{(0)}, q^{(0)}) \right\},
\]

(2.32)

where the dimensionless tensor \( \mathbf{\hat{D}}_e \) of extracellular conductivity is given by:

\[
\mathbf{\hat{D}}_e = \frac{\mu}{|Y_e|} \left\{ |Y_e| I - \int_S \mathbf{n}_e \otimes \mathbf{w}_e \, ds \right\},
\]

\( \mu \) is the ratio of extracellular and intracellular conductivities, defined in (2.14) and \( \mathbf{w}_e = (w^1_e(\xi), w^2_e(\xi), w^3_e(\xi))^T \) and its components \( w^k_e(\xi) \) \( (k = 1, 2, 3) \), satisfy

\[
\Delta_{\xi\xi} w^k_e(\xi) = 0, \quad \text{in} \ Y_e, \quad (2.33a)
\]

\[
\nabla_{\xi} w^k_e(\xi) \cdot \mathbf{n}_e = n_{e, \xi_k} \quad \text{on} \ S, \quad (2.33b)
\]

where \( \mathbf{n}_e = (n_{e, \xi_1}, n_{e, \xi_2}, n_{e, \xi_3})^T \) is the unit outward normal to the membrane portion \( S \) with respect to the extracellular portion \( Y_e \). The problems for \( w^k_e(\xi) \) \( (k = 1, 2, 3) \) are solvable up to a constant, and the solutions can be fixed, e.g., by the condition \( \int_S w^k_e(\xi) \, d\xi = 0 \).

Using the scalings (2.11)-(2.12) and the definitions of macroscopic time and space units \( T \) and \( L \), we can rescale the dimensionless equations (2.31)-(2.32) to the following dimensional averaged equations for the macroscopic intra- and extracellular...
potentials:
\[
\nabla_x \cdot (D_i \nabla_x \Phi_i) = \beta \left\{ C_m \frac{\partial}{\partial t} V_m + I_{ion}(V_m, q) \right\},
\]
(2.34a)
\[
\nabla_x \cdot (D_e \nabla_x \Phi_e) = -\beta \left\{ C_m \frac{\partial}{\partial t} V_m + I_{ion}(V_m, q) \right\},
\]
(2.34b)

where the dimensional surface-to-volume ratio \( \beta = \hat{\beta}/d \) and the dimensional conductivity tensors \( D_i \) and \( D_e \) are given by

\[
D_i = \sigma_i \hat{D_i} \quad \text{and} \quad D_e = \sigma_e \hat{D_e}.
\]

Obviously, the identities (2.34) together with the nonlinear reactions for state variables \( q \) exactly make up the macroscopic bidomain model (2.1).

2.1.3 Anisotropic Conductivity

It is easy to verify that the conductivity tensors \( D_i \) and \( D_e \) are symmetric and positive definite [53, 82]. Hence, there are three eigenvalues and three orthogonal eigenvectors for each of the conductivity tensors even though they depend on the position \( x \).

Actually, the ventricular myocardium may be conceived as a set of muscle sheets running radially from epicardium to endocardium [113] and the three eigenvectors of the intra- or extracellular conductivity tensor at any position correspond to the three principal directions of the cardiac microstructure [135]: one along the myocardial fiber (\( e_l(x) \)), a second orthogonal to the fiber direction and lying in the myocardial sheet plane (\( e_t(x) \)) and a third orthogonal to the first two in the cross-sheet direction (\( e_n(x) \)). In other words, \( e_l(x) \) parallels to the local fiber direction; \( e_t(x) \) and \( e_n(x) \) are tangential and normal to the muscle sheet, respectively. The effective conductivities of intra- and extracellular spaces always have the same principal axes: \( e_l(x) \), \( e_t(x) \) and \( e_n(x) \) [133].
Based on the arguments above, the intra- and extracellular conductivity tensors \( D_i \) and \( D_e \) may be given by

\[
D_i = \sigma^{(l)}_i e_i(x) e_i^T(x) + \sigma^{(t)}_i e_t(x) e_t^T(x) + \sigma^{(n)}_i e_n(x) e_n^T(x), \quad (2.35a)
\]

\[
D_e = \sigma^{(l)}_e e_i(x) e_i^T(x) + \sigma^{(t)}_e e_t(x) e_t^T(x) + \sigma^{(n)}_e e_n(x) e_n^T(x). \quad (2.35b)
\]

Sometimes, the expression for the intra- and extracellular conductivity tensors can be simplified by assuming that the transverse coupling is the same in all angular directions orthogonal to the fiber axis (\( \sigma^{l,t,e}_{i,e} = \sigma^{n}_{i,e} \)) \([76, 141]\). Note that the three principal directions \( e_l(x), e_t(x) \) and \( e_n(x) \) are orthogonal to each other. The \( x \) coordinate system can be locally oriented such that

\[
e_l(x)e_l^T(x) + e_t(x)e_t^T(x) + e_n(x)e_n^T(x) = I, \quad (2.36)
\]

where \( I \) is the \( 3 \times 3 \) identity matrix. Substituting (2.36) into (2.35a)-(2.35b), we recover the axially isotropic conductivity tensors:

\[
D_i = \sigma^{(n)}_i I + (\sigma^{(l)}_i - \sigma^{(n)}_i) e_i(x) e_i^T(x) \quad (2.37a)
\]

\[
D_e = \sigma^{(n)}_e I + (\sigma^{(l)}_e - \sigma^{(n)}_e) e_t(x) e_t^T(x). \quad (2.37b)
\]

Generally, the principal conductivity \( \sigma^{l,t,n}_{i,e} \) in each direction is not equal. Hence, the conductivity must be anisotropic. The anisotropy ratio of experimentally measured effective conductivities varies in the range from 5.7 to 10.8 \([59, 133, 157]\).

The anisotropic conductivity is closely related to the fiber orientation of the cardiac tissue. The main direction of the anisotropy changes with the depth of myocardium. It can rotate up to 120° or more through the myocardial wall \([100, 147]\). The rotational anisotropy creates quite different conditions for wave propagation in three dimensions and presents a great challenge for numerical simulation as well \([141]\).
2.2 The Simplified Monodomain Model

In the one-dimensional case, the conductivity tensors \( \mathbf{D}_i \) and \( \mathbf{D}_e \) degenerate to scalar values \( \sigma_i \) and \( \sigma_e \). In the case that \( \sigma_i \) and \( \sigma_e \) are constants, we can reduce the number of equations in the macroscopic bidomain model (2.34) from two to one by multiplying (2.34a) by \( \frac{\sigma_e}{\sigma_i + \sigma_e} \), multiplying (2.34b) by \( \frac{\sigma_i}{\sigma_i + \sigma_e} \) and subtracting them. After that, we can get the following equation for the transmembrane potential \( V_m = \Phi_i - \Phi_e \):

\[
\nabla \cdot \left( \frac{\sigma_i \sigma_e}{\sigma_i + \sigma_e} \nabla V_m \right) = \beta \left\{ C_m \frac{\partial V_m}{\partial t} + I_{ion}(V_m, q) \right\}.
\]

(2.38)

However, for two and three space dimensional cases, the ratios of directional conductivities in the intra- and extracellular spaces are generally not equal. The corresponding simplification is usually not available unless we assume that the intra- and extracellular conductivity tensors have the same constant anisotropy ratios, i.e.,

\[
\frac{\sigma_i^{(l)}}{\sigma_e^{(l)}} = \frac{\sigma_i^{(t)}}{\sigma_e^{(t)}} = \frac{\sigma_i^{(n)}}{\sigma_e^{(n)}} \equiv \lambda.
\]

For cardiac tissue, the constant \( \lambda \) is usually assumed to be spatially uniform. In other words, \( \lambda \) is a fixed constant.

In the case of equal anisotropy ratio of the intra- and extracellular spaces, we assume \( \mathbf{D}_i = \lambda \mathbf{D}_e \) for the conductivity tensors (2.35a) and (2.35b) and set \( \mathbf{D} = \frac{\lambda \mathbf{D}_i}{1 + \lambda} \). Then the bidomain system (2.1) can be reduced to the anisotropic monodomain model involving only a reaction-diffusion equation for the transmembrane potential \( V_m \):

\[
\nabla \cdot \mathbf{D} \nabla V_m = \beta \left\{ C_m \frac{\partial V_m}{\partial t} + I_{ion}(V_m, q) \right\},
\]

(2.39a)

\[
\frac{\partial q}{\partial t} = \mathcal{M}(V_m, q).
\]

(2.39b)

In addition to the variable dimension reduction, the greatest difference of the
monodomain model (2.39) from the bidomain model (2.1) lies in the capacitance matrix, which is the coefficient of the first derivative of time.

As noted, in the bidomain model, the capacitance matrix (2.2) is singular. Due to the singularity of the capacitance, the bidomain model (2.1) is actually a degenerate parabolic reaction-diffusion system or a system of elliptic-parabolic equations like (2.3). An elliptic solver involving a matrix inversion must be employed to numerically integrate those equations. This usually requires a huge amount of computer memory and long computer time.

However, in the monodomain model, the capacitance matrix degenerates to a scalar positive number

\[ C_m = C_m. \]  (2.40)

The equation (2.39a) is a normal reaction-diffusion equation. A cheap numerical method can always be used although there are some other restrictions. The monodomain model gains its popularity due to the less requirement on computational resources.

Nevertheless, the monodomain model is not adequate since it is unable to reproduce some patterns and morphology of the experimentally observed extracellular potential maps and electrograms. That is, to explain certain important features of the electrocardiogram, the cardiac tissue must be viewed as a bidomain with unequal anisotropy ratios [72, 74, 77, 78, 93, 141, 175].

### 2.3 Cardiac Models for Membrane Dynamics

Note that a current-voltage relationship \( I_{ion}(V_m, q) \) and corresponding nonlinear dynamics \( M(V_m, q) \) must be supplied to give a complete description of the bidomain and monodomain models. An instance of \( I_{ion}(V_m, q) \) and \( M(V_m, q) \) is called a mem-
brane model. It represents the dynamics of a single cell. In the literature, there are a variety of membrane models, such as the Beeler-Reuter model [19], the Luo-Rudy models [120, 121], the Rogers-McCulloch model [158], the Fenton-Karma model [70] and the recent Schaeffer models [129, 164].

The first membrane model is the one proposed by Hodgkin and Huxley for the nerve action potentials [95], which earned them the Nobel prize in Medicine in 1963. In the model, the ionic current $I_{\text{ion}}(V_m, \mathbf{q})$ through channels of the membrane is a nonlinear function of the transmembrane potential $V_m$ and the state variables $\mathbf{q} = (q_1, q_2, q_3)^T \in \mathbb{R}^3$. The dynamics corresponding to each state variable $q_j$ has a special structure like the following:

$$\frac{\partial q_j}{\partial t} = \alpha_j(V_m)(1 - q_j) - \beta_j(V_m)q_j,$$

with $\alpha_j(V_m), \beta_j(V_m) > 0$ and $0 \leq q_j \leq 1$. Here, $\alpha_j(V_m)$ and $\beta_j(V_m)$ are nonlinear functions of the transmembrane potential $V_m$. For a detailed description of the nonlinear dynamics of the Hodgkin-Huxley model, refer to the original paper [95].

Membrane models of Hodgkin-Huxley type have been later developed for the cardiac action potentials. They are becoming more and more complex due to the improved experimental data over the years. Such models include the Beeler-Reuter model [19], the phase-I Luo-Rudy [120] the phase-II Luo-Rudy models [120, 121], and even the more complicated models, such as the Luo-Rudy dynamic model and the Winslow model [192]. The first two models involve six gating variables, one ionic concentration variable and membrane dynamics of the Hodgkin-Huxley type while the latter two models involve around thirty state variables and very complicated membrane dynamics.

Many membrane models in the literature belong to the simplified membrane models. Some models, called eikonal equations, just describe the motion of the excitation
wave fronts [20, 77, 106]. Some models are not based on any experimentally measured quantities such as the the simplest and most widely used FitzHugh-Nagumo (FHN) model [71].

The FHN model has only one gating variable. In the thesis work, we use the following form of the FHN model

\[ I_{\text{ion}}(V_m, q) = -\lambda \{ q - V_m(V_m - \theta)(1 - V_m) \}, \]
\[ \mathcal{M}(V_m, q) = \alpha V_m - \beta q, \]

which is dimensionless. In the model, the eigenvalue parameter \(|\lambda| >> 1\) is negative. The threshold parameter \(\theta \in (0, \frac{1}{2})\), the other two parameters \(\alpha, \beta\) are positive constants such that \(\beta = O(1)\) and \(\alpha > \frac{1}{2}\beta(1 - \theta)^2\). This inequality guarantees that the only stationary state of the reaction in the limit of vanishing diffusion is \((V_m, q) = (0, 0)\) and the traveling wave has positive speed [182]. Usually, the conductivity tensor for the FHN model is replaced by a small positive scalar constant \(\epsilon << 1\).

While the FHN model is not based on any experimentally measured quantities, it exhibits many phenomena found in a wide range of excitable systems. The FHN model is well-studied and some observations in its numerical simulations have been well documented. This can provide insights into the general dynamics of excitable systems and may help us validate the code being developed in this thesis project.

There are also ionically based simplified models [70, 129, 164]. For example, the Mitchell-Schaeffer membrane model [129] contains two currents but has one state variable only. It has the following form

\[ I_{\text{ion}}(V_m, q) = \frac{\Delta V}{R_m} \left\{ \frac{V_m/\Delta V}{\tau_{\text{out}}} - \frac{(V_m/\Delta V)^2(1 - V_m/\Delta V)}{\tau_{\text{in}}} q \right\}, \]
\[ \mathcal{M}(V_m, q) = \frac{1}{R_mC_m} \frac{q_{\infty}(V_m/\Delta V) - q}{\tau_{\infty}(V_m/\Delta V)}, \]
where \( R_m \) is the surface resistivity of the membrane that separates the inside and the outside of cells. Also, \( \Delta V \) is a unit of measure for \( \Phi_i, \Phi_e \) and \( V_m \) in the bidomain equations (2.1). Typical values of \( R_m \) and \( \Delta V \) are \( 2 \cdot 10^4 \) ohms cm\(^2\), 100 mV, respectively [133]. The dimensionless time constant \( \tau_{\infty}(v_m) \) and the state variable constant \( q_{\infty}(v_m) \) are defined by

\[
\tau_{\infty}(v) = \begin{cases} 
\tau_{\text{open}}, & v_m < v_{\text{gate}} \\
\tau_{\text{close}}, & v_m \geq v_{\text{gate}}
\end{cases} \quad \text{and} \quad q_{\infty}(v) = \begin{cases} 
1, & v_m < v_{\text{gate}} \\
0, & v_m \geq v_{\text{gate}}
\end{cases}.
\]

The positive parameters are based on the assumption that \( \tau_{\text{in}} \ll \tau_{\text{out}} \ll \tau_{\text{open}}, \tau_{\text{close}} \), and \( 0 < v_{\text{gate}} \ll 1 \). For instance, we may choose: \( \tau_{\text{in}} = 0.005, \tau_{\text{out}} = 0.1, \tau_{\text{open}} = 1.5, \tau_{\text{close}} = 7.5 \) and \( v_{\text{gate}} = 0.1 \).

As concluded by Fenton [69], simplified models of cardiac dynamics have been proved to be valuable in the study of arrhythmias when extensive and time consuming simulations are required. It allows us to vary parameters and study different levels of complexity in a shorter time. Results then can be tested with just key simulations using the more complex models.

### 2.4 Reaction-Diffusion Systems

Both the bidomain model (2.1) and the monodomain model (2.39) are reaction-diffusion systems. In effect, they are singularly perturbed systems for physiologically realistic choices of the model parameters and reasonable initial data.

It is well-known that the singularly perturbed systems exhibits threshold and traveling wave phenomena for appropriate model parameters and data [46, 84, 105, 168, 194]. By a threshold phenomenon, we mean that there is a threshold value of the transmembrane potential \( V_m \), for example in the cardiac model (2.1) or (2.39) in the spatially uniform case, below which a quick return to the steady state occurs and
above which there is a large excursion before an eventual return to the steady state. The threshold phenomenon leads to stable traveling wave solutions when diffusion is included in the system. In the cardiac tissue, electrical waves are triggered by a sufficiently large initial perturbation of the transmembrane potential $V_m$ from the steady state, and after their passage the tissue again becomes excitable and amenable to the triggering of another electrical wave.

For some simplified model problems such as the FitzHugh-Nagumo model (2.41) and the Mitchell-Schaeffer model (2.42), it is possible to use an asymptotic analysis technique to estimate the front speed and analyze traveling waves [33, 73]. It is noteworthy that the estimation of wave speed may help us determine the time step size to be used in a numerical scheme. In particular, in our adaptive mesh refinement (AMR) algorithm, it is useful to guarantee that the waves travel at most one cell per time step.

However, the general reaction-diffusion systems arising from modeling the electrical activity of the heart involve complicated nonlinear reactions for the state variables $\mathbf{q}$ and the spatial diffusion in the systems is generally anisotropic and inhomogeneous (even heterogeneous). It is very difficult, if not impossible, to make an analytical study for them [20, 33, 73, 75]. Instead, numerical simulations should be performed.
Chapter 3

Numerical Methods

In this chapter, numerical methods for the singularly perturbed reaction-diffusion systems arising from modeling the electrical activity in the heart will be discussed in detail. First, operator splitting techniques, including the first-order Godunov splitting and the second-order Strang splitting, for the reaction-diffusion systems are presented in section 3.2. The application of an operator splitting scheme to a reaction-diffusion system results in spatially independent nonlinear stiff reactions and spatially dependent diffusion. In section 3.3, a second-order singly diagonally implicit Runge-Kutta (SDIRK) method, which is both absolutely stable (A-stable) and large eigenvalue stable (L-stable) for the stiff nonlinear reactions, is introduced. Finally, we describe second-order conforming finite element discretizations of the resulting linear diffusion on both uniformly and locally refined grids in section 3.4. The discretization on locally refined grids yields composite grid equations.

3.1 Introduction

From Chapter 2, we know that the electrical activity in the heart can be modeled by a partial differential system coupled with a set of ordinary differential equations. In the bidomain model, the partial differential equations make up a degenerate parabolic system:

\[
\nabla \cdot D_i \nabla \Phi_i = \beta \left\{ C_m \frac{\partial V_m}{\partial t} + I_{ion}(V_m, q) \right\}, \tag{3.1a}
\]

\[
\nabla \cdot D_e \nabla \Phi_e = -\beta \left\{ C_m \frac{\partial V_m}{\partial t} + I_{ion}(V_m, q) \right\}. \tag{3.1b}
\]
In the simplified monodomain model, the partial differential equations are reduced to a typical singularly perturbed reaction-diffusion equation:

$$\nabla \cdot (D \nabla V_m) = \beta \left\{ C_m \frac{\partial V_m}{\partial t} + I_{ion}(V_m, q) \right\},$$  \hspace{1cm} (3.2)

If the state variables $q$ are considered as parameters, both sets of partial differential equations (3.1) and (3.2) can be written in a general form as follows:

$$C \frac{\partial u}{\partial t} = \nabla \cdot D \nabla u + R(u),$$  \hspace{1cm} (3.3)

or typically,

$$C \frac{\partial u}{\partial t} = \epsilon \nabla \cdot \nabla u + \lambda u.$$  \hspace{1cm} (3.4)

Here, $u$ and $R(u)$ are a vector and a vector-valued function, respectively. The coefficient $C$ of the first order time derivative is a constant corresponding to the monodomain model, or a singular matrix corresponding to the bidomain model. The spatially dependent diffusion tensor $D \equiv D(x)$ is symmetric positive definite, continuous in the domain $\Omega$ and has uniform positive lower and upper bounds. The nonlinear vector-valued function $R(u)$ is assumed to be continuously differentiable. The small positive parameter $\epsilon \ll 1$ in (3.4) stands for a typical scale of the diffusion tensor $D(x)$. The large negative parameter $|\lambda| >> 1$ in (3.4) represents a typical scale of the eigenvalues of the Jacobian matrix $\frac{\partial R(u)}{\partial u}$ of the nonlinear reaction $R(u)$ in (3.3). Both parameters $\epsilon$ and $\lambda$ in (3.4) emphasize that the reaction-diffusion system (3.3) is singularly perturbed, due to slow diffusive and fast reactive cardiac tissue.

### 3.2 Operator Splitting for the Reaction-diffusion Systems

To numerically solve the reaction-diffusion equations (3.3), it is useful to treat implicitly both the diffusion and reaction terms since there are always time step restrictions.
associated with explicit schemes, even if only one of the reaction or diffusion term is explicitly integrated. However, when we do so, we must provide a nonlinear solver for a large nonlinear system in every time step. This motivates us to seek for a strategy, with which both of the reaction and diffusion terms are integrated implicitly but there is no need to solve large nonlinear systems.

Operator splitting is such a method, which is widely used in science and engineering applications [17, 47, 99, 101, 128, 169, 170, 197]. It allows us to split the diffusion from the reactions and integrate them independently and implicitly. The resulting nonlinear reactions are spatially decoupled. Hence, the size of nonlinear systems is greatly reduced, and the large linear systems for the diffusion are self-adjoint.

### 3.2.1 First-order Splitting

The first-order splitting separates the bidomain reaction-diffusion system into a linear diffusion and a set of spatially decoupled nonlinear reactions. This splitting is also called the Godunov splitting [123].

(I). The linear diffusion part:

\[
\nabla \cdot D_i \nabla \Phi_i = \beta C_m \frac{\partial V_m}{\partial t} \quad \text{for} \ t^n < t < t^{n+1}, \quad (3.5a)
\]

\[
\nabla \cdot D_e \nabla \Phi_e = -\beta C_m \frac{\partial V_m}{\partial t} \quad \text{for} \ t^n < t < t^{n+1}. \quad (3.5b)
\]

Here, the transmembrane potential \( V_m \) is defined as the difference of intra- and extracellular potentials: \( V_m = \Phi_i - \Phi_e \).

(II). The nonlinear reactions part:

\[
C_m \frac{\partial V_m}{\partial t} + I_{ion}(V_m, q) = 0 \quad \text{for} \ t^n < t < t^{n+1}, \quad (3.6a)
\]

\[
\frac{\partial q}{\partial t} = M(V_m, q) \quad \text{for} \ t^n < t < t^{n+1}. \quad (3.6b)
\]
We first integrate the linear diffusion part by a full time step; then, using the advanced data from the diffusion as initial value, integrate the nonlinear reactions by another time step.

For the monodomain model (2.39), the linear diffusion resulting from operator splitting is replaced by a scalar equation:

$$\nabla \cdot \mathbf{D} \nabla V_m = \beta C_m \frac{\partial V_m}{\partial t} \quad \text{for} \quad t^n < t < t^{n+1}. \quad (3.7)$$

3.2.2 Second-order Splitting

The second-order splitting separates the bidomain reaction-diffusion system (2.1) into a linear diffusion and a set of nonlinear reactions too, but the decoupled parts are integrated in a symmetric way, where the first-order temporal truncation errors are canceled. In the literature, this splitting is called the Strang splitting [172], which has been employed by Joakim Sundes [173] and Qu et al. [154] to simulate the electrical activity in the heart.

(I). First half time step nonlinear reaction:

$$C_m \frac{\partial V_m}{\partial t} + I_{ion}(V_m, q) = 0 \quad \text{for} \quad t^n < t < t^n + \frac{\Delta t}{2}, \quad (3.8a)$$

$$\frac{\partial q}{\partial t} = \mathcal{M}(V_m, q) \quad \text{for} \quad t^n < t < t^n + \frac{\Delta t}{2}. \quad (3.8b)$$

(II). A full time step linear diffusion:

$$\nabla \cdot \mathbf{D}_i \nabla \Phi_i = \beta C_m \frac{\partial V_m}{\partial t} \quad \text{for} \quad t^n < t < t^{n+1}, \quad (3.9a)$$

$$\nabla \cdot \mathbf{D}_e \nabla \Phi_e = -\beta C_m \frac{\partial V_m}{\partial t} \quad \text{for} \quad t^n < t < t^{n+1}. \quad (3.9b)$$
(III). Second half time step nonlinear reaction:

\[ C_m \frac{\partial V_m}{\partial t} + I_{\text{ion}}(V_m, q) = 0 \quad \text{for } t^n + \frac{\Delta t}{2} < t < t^{n+1}. \] (3.10a)

\[ \frac{\partial q}{\partial t} = \mathcal{M}(V_m, q) \quad \text{for } t^n + \frac{\Delta t}{2} < t < t^{n+1}. \] (3.10b)

We first integrate the reactions by a half time step; using the advanced data from the reactions as initial data, integrate the linear diffusion by a full time step; finally, using the results from the diffusion problem as initial data, integrate the nonlinear reactions by another half time step.

For the monodomain model (2.39), similar to the first-order splitting case, the linear diffusion resulting from operator splitting is replaced by a scalar equation:

\[ \nabla \cdot D \nabla V_m = \beta C_m \frac{\partial V_m}{\partial t} \quad \text{for } t^n < t < t^{n+1}. \] (3.11)

**Remark 3.1:** In the two splittings above, we always put the stiff nonlinear reactions at the end of the split step since, for the cardiac problems, the stiffness mostly comes from the reaction terms when the grid size is not extremely small. It is known that putting the stiff computation at the end enhances accuracy [169].

### 3.3 Time Integration for Stiff Nonlinear Reactions

Note that the nonlinear reactions resulting from operator splitting applied to either the bidomain model problem (2.1) or the monodomain model problem (2.39) have the following form:

\[ \frac{d\mathbf{u}(t)}{dt} = \mathbf{R}(\mathbf{u}(t)) \quad \text{for } t^n < t < t^{n+1}, \] (3.12a)

\[ \mathbf{u}(t^n) = \mathbf{u}^n, \] (3.12b)
where \( \mathbf{u}(t) \) is a vector and \( \mathbf{R}(\mathbf{u}(t)) \) is a vector-valued function. In the initial value problem (3.12), the vector-valued function \( \mathbf{R}(\mathbf{u}(t)) \) involves at least two scales: a fast scale and a slow scale, which makes the problem stiff. The nonlinear reactions (3.12) above are stiff ordinary differential equations.

In the literature, there are already many excellent results on integration schemes for stiff ordinary differential equations (ODE) or differential algebraic equations [49, 89, 111, 166]. For an investigation of different solvers for stiff ODE systems arising from modeling the electrical activity in the heart, refer to [173]. In the following subsections, we only briefly discuss the schemes that are directly related to our thesis work.

### 3.3.1 A-Stable Schemes

Let us first consider the simple linear ordinary differential equation (ODE)

\[
\frac{d\mathbf{u}(t)}{dt} = \lambda \mathbf{u}(t). \tag{3.13}
\]

Here, \( \lambda \) is a complex number, which mimics eigenvalues of more complicated ordinary differential equations. Suppose an one-step numerical scheme is employed to integrate the ordinary differential equation: given an approximation \( \mathbf{u}^n \) to the true solution at \( t^n \), the numerical value of \( \mathbf{u}(t) \) at \( t^{n+1} = t^n + \Delta t \) is computed by

\[
\mathbf{u}^{n+1} = S(\lambda \Delta t)\mathbf{u}^n, \tag{3.14}
\]

where the function \( S(z) \) is the so-called stability function.

The stability function \( S(z) \) of the forward Euler scheme for the simple ODE (3.13) has the form

\[
S_l(z) = 1 + z. \tag{3.15}
\]
The stability function $S(z)$ of the backward Euler scheme for the simple ODE (3.13) reads

$$S_r(z) = \frac{1}{1-z}. \quad (3.16)$$

The stability function $S(z)$ of the trapezoidal scheme for the simple ODE (3.13) is given by

$$S_c(z) = \frac{1 + z/2}{1 - z/2}. \quad (3.17)$$

Denote by $e^n = u(t) - u^n$ the approximation error at $t^n$. Then, the approximation error propagates by

$$e^{n+1} = S(\lambda \Delta t)e^n + LTE \approx S(\lambda \Delta t)e^n.$$  

Here, LTE represents the local truncation error, which is on the order of $\Delta t^2$ for the Euler schemes and on the order of $\Delta t^3$ for the Trapezoidal method. The one-step scheme (3.14) is said to be stable if the approximation error $e^n$ never blows up or goes unbounded. That is, the stability function satisfies

$$|S(z)| \leq 1. \quad (3.18)$$

The stability region for the scheme (3.14) is defined by

$$\mathcal{D} = \mathcal{D}(S) \equiv \{ z \in \mathbb{C} : |S(z)| \leq 1 \}, \quad (3.19)$$

where $\mathbb{C}$ stands for the whole complex plane.

It is easy to find that the stability region $\mathcal{D}_f = \mathcal{D}(S_f)$ of the forward Euler method is very narrow. It is the unit circle centered at $z = -1$ on the complex plane. The backward Euler method is stable as long as $|S_r(z)| < 1$ or $|1-z| \geq 1$. Its stability region $\mathcal{D}_r = \mathcal{D}(S_r)$ almost covers the whole complex plane except the unit circle centered at $z = 1$. The trapezoidal method is stable as long as the real part of $z$ is non-positive, i.e., its stability region $\mathcal{D}_c = \mathcal{D}(S_c)$ covers the left half complex plane.
For most engineering problems, we deal with cases where the real part of the eigenvalue \( \lambda \) in (3.13) is negative. Therefore, there is no stability restriction for the backward Euler and the trapezoidal methods at all. In this sense, we say these two methods are \textit{absolutely stable} on the left half part of the complex plane, or simply \textit{A-stable}. Formally, a numerical integration scheme is said to be \textit{A-stable} if the stability region \( D \) of the method covers the entire half complex plane around the negative real axis for the simple linear ordinary differential equation [63]. An A-stable scheme has no stability restriction for most engineering problems.

It is known that an explicit scheme such as the forward Euler method usually has only a finite and bounded stable region even for the simple linear ODE (3.13) while an implicit scheme may have an infinite and unbounded stable region. In the literature, it is also concluded that an A-stable scheme must not be a multi-step method and an A-stable scheme with order greater than two must not be a linear method [63].

### 3.3.2 L-Stable Schemes

In this subsection, we discuss the accuracy of an A-stable scheme for problems with large eigenvalues and restrict ourselves only to the case that the eigenvalue \( \lambda \in \mathbb{C} \) in the simple linear ODE (3.13) is in the left half part of the complex plane \( \mathbb{C} \).

Notice that, given an approximation \( u^n \) of the solution \( u(t^n) \) at \( t^n \), the local exact solution to the the simple linear ODE (3.13) at \( t^{n+1} = t^n + \Delta t \) is given by

\[
\bar{u}(t^{n+1}) = e^{\lambda \Delta t} u^n.
\]

(3.20)

We can compute the local error of the numerical solution by

\[
\bar{e}^{n+1} \equiv \bar{u}(t^{n+1}) - u^{n+1} = (e^{\lambda \Delta t} - S(\lambda \Delta t)) u^n,
\]

(3.21)

which is a measure of the accuracy of the numerical scheme used. An “accurate” A-stable scheme should generate small local errors even for problems with large
eigenvalues. In other words, with a fixed time step $\Delta t$, the local error $\tilde{e}^{n+1}$ of an "accurate" A-stable scheme is supposed to approach to zero as the modulus of the eigenvalue $\lambda$ goes to infinity. That is, it is desirable that the amplification factor $e^{\lambda \Delta t} - S(\lambda \Delta t)$ in (3.21) satisfies

$$\lim_{Re(z) < 0, |z| \to \infty} |e^z - S(z)| = 0,$$

or

$$\lim_{Re(z) < 0, |z| \to \infty} |S(z)| = 0. \quad (3.22)$$

We say an A-stable scheme whose stability function $S(z)$ satisfies the property (3.22) is a stiffly accurate A-stable scheme, or simply an L-stable scheme [89].

It is easy to find that, the stability function $S_r(z)$ of the backward Euler method satisfies

$$\lim_{Re(z) < 0, |z| \to \infty} |S_r(z)| = 0,$$

and the property (3.22) does not hold for the stability function $S_c(z)$ of the trapezoidal method, i.e.,

$$\lim_{Re(z) < 0, |z| \to \infty} |S_c(z)| = 1.$$ 

So, the backward Euler scheme is L-stable while the trapezoidal is not. Note that the backward Euler method has only first-order accuracy.

The words "stiffly accurate" in the definition of an L-stable scheme come from the following fact. Some engineering problems, such as those from computational cardiology, have a wide range of eigenvalues over the complex plane. The appearance of both large and small eigenvalues in the problems makes them "stiff". An L-stable scheme guarantees that accurate results are computed for the stiff components of these problems, using time step sizes that are appropriate for less-stiff components.
3.3.3 The Singly Diagonally Implicit Runge-Kutta Method

In this section, we will describe in detail a second-order L-stable scheme. It belongs to the singly diagonally implicit Runge-Kutta methods (SDIRK) [48, 49, 65, 89].

For integrating the ordinary differential equations (3.12), the second-order SDIRK scheme takes the following form

\[ k_1 = R(u^n + \gamma k_1 \Delta t), \]  
\[ k_2 = R(u^n + (1 - 2\gamma)k_1 \Delta t + \gamma k_2 \Delta t), \]  
\[ u^{n+1} = u^n + \frac{k_1 + k_2}{2} \Delta t. \]  

(3.23a)  
(3.23b)  
(3.23c)

Here, the constant \( \gamma \) is a parameter to be determined. Since the function \( R(u) \) is nonlinear, the stage vectors \( k_1 \) and \( k_2 \) are defined implicitly. A nonlinear solver must be invoked to compute their values in the implementation.

To analyze the scheme, we assume the two vectors \( k_1 \) and \( k_2 \) have the following forms

\[ k_1 = k_1^{(0)} + k_1^{(1)} \gamma \Delta t + k_1^{(2)} (\gamma \Delta t)^2 + \cdots, \]  
\[ k_2 = k_2^{(0)} + k_2^{(1)} \gamma \Delta t + k_2^{(2)} (\gamma \Delta t)^2 + \cdots. \]  

(3.24a)  
(3.24b)

By substituting the expansions (3.24) above into the scheme (3.23) and using the Taylor expansion of \( R(u) \) around \( u = u^n \), we can easily find the coefficients in the expansions (3.24) for those two stage vectors, and their approximate values

\[ k_1 \approx R + R'R \gamma \Delta t + \left\{ (R')^2 R + \frac{1}{2} R^T R'' R \right\} (\gamma \Delta t)^2, \]
\[ k_2 \approx R + \frac{1 - \gamma}{\gamma} R'R \gamma \Delta t + \left\{ \frac{2 - 3\gamma}{\gamma} (R')^2 R + \frac{1}{2} \left( \frac{1 - \gamma}{\gamma} \right)^2 R^T R'' R \right\} (\gamma \Delta t)^2. \]

Here, we assume the function \( R \) is at least twice continuously differentiable. The function \( R \), its Jacobian \( R' \) and Hessian \( R'' \) are all evaluated at \( u^n \). Finally, the
computed solution has the following approximation

\[ u^{n+1} \approx u^n + R \Delta t + \frac{1}{2} R' R \Delta t^2 + \gamma (1 - \gamma) (R')^2 R \Delta t^3 + \frac{2 \gamma^2 - 2 \gamma + 1}{4} R^T R'' R \Delta t^3. \]

On the other hand, a Taylor expansion of the true solution \( u(t^{n+1}) \) around \( t^n \) gives

\[ u(t^{n+1}) \approx u(t^n) + R \Delta t + \frac{1}{2} R' R \Delta t^2 + \frac{1}{6} (R')^2 R \Delta t^3 + \frac{1}{6} R^T R'' R \Delta t^3. \]

Comparing the two approximations above gives an estimate of the local truncation error for the scheme (3.23), i.e.,

\[ \epsilon_{sdirk2} \approx \left\{ \frac{1}{6} - \gamma (1 - \gamma) \right\} (R')^2 R \Delta t^3 + \left\{ \frac{1}{6} - \frac{2 \gamma^2 - 2 \gamma + 1}{4} \right\} R^T R'' R \Delta t^3. \quad (3.26) \]

It is obvious that, for any value of the parameter \( \gamma \), the local truncation error \( \epsilon_{sdirk2} \) is at least on the order of \((\Delta t)^3\). Hence, the scheme (3.23) at least has second-order accuracy for any \( \gamma \). In fact, the scheme has third-order accuracy if the parameter \( \gamma \) is selected such that

\[
\frac{1}{6} - \gamma (1 - \gamma) = 0,
\]

\[
\frac{1}{6} - \frac{2 \gamma^2 - 2 \gamma + 1}{4} = 0,
\]

which have two solutions \( \gamma = (3 \pm \sqrt{3})/6 \). However, these parameters do not have the stability properties desired.

In order to study the A- and L-stabilities of the second-order SDIRK scheme (3.23), we apply it to the simple linear ODE (3.13). Straightforward computation shows that the corresponding stability function \( S(z) \) for this scheme is given by

\[ S(z) = \frac{1 + (1 - 2 \gamma) z + (\gamma^2 - 2 \gamma + 1/2) z^2}{(1 - \gamma z)^2}. \]
By the definition of L-stability, the scheme (3.23) is L-stable only if the limit (3.22) is equal to zero or the coefficient of the quadratic term $z^2$ in the numerator of $S(z)$ vanishes, i.e.,

$$
\gamma^2 - 2\gamma + 1/2 = 0.
$$

This gives the parameters $\gamma = 1 \pm \sqrt{2}/2$, which are the only possible choices for the scheme to be L-stable. In fact, with either of the parameters, the scheme (3.23) is A-stable and hence L-stable.

With $\gamma = 1 \pm \sqrt{2}/2$, the stability function $S(z)$ becomes

$$
S(z) = \frac{1 + (1 - 2\gamma)z}{(1 - \gamma z)^2}.
$$

(3.28)

Note that the linear fractional transformation

$$
w(z) = \frac{2\gamma z}{1 - \gamma z}
$$

maps the left half-plane onto the unit circle centered at $w = -1$ since

$$
|w(z) + 1| = \frac{|1 + \gamma z|}{|1 - \gamma z|} \leq 1 \quad \text{for } z \in \mathbb{C}, \quad \text{Re}(z) \leq 0.
$$

Using this mapping, we can transform the stability function (3.28) into the new form

$$
S(w) = \frac{w + 2}{4} \left\{ \frac{1}{\gamma} - 1 \right\} (w + 1) + \left( 3 - \frac{1}{\gamma} \right)
$$

$$
= \frac{w + 2}{4} \left\{ \frac{1 - \gamma}{\gamma} (w + 1) - \frac{\gamma}{1 - \gamma} \right\}
$$

in terms of $w \in \mathbb{C}$. By the maximum modulus principle, the stability function $S(w)$ is analytical and the maximum value of $|S(w)|$ on the unit circle centered at $w = -1$ must occur on the boundary of the circle. Now, let

$$
w = \cos(\theta) + i \sin(\theta) - 1
$$

be a point on the boundary. The stability function $S(w)$ becomes

$$S(\theta) = \frac{1 + \cos(\theta) + i \sin(\theta)}{4} \left\{ \frac{1 - \gamma}{\gamma} (\cos(\theta) + i \sin(\theta)) - \frac{\gamma}{1 - \gamma} \right\}$$

The square of the modulus of $S(\theta)$ is then given by

$$|S(\theta)|^2 = \frac{1 + \cos(\theta)}{8} \left\{ \left( \frac{1 - \gamma}{\gamma} \right)^2 + \left( \frac{\gamma}{1 - \gamma} \right)^2 - 2 \cos(\theta) \right\}$$

$$= \frac{1 + \cos(\theta)}{8} \{ 6 - 2 \cos(\theta) \}$$

$$= \frac{1 + \cos(\theta)}{2} \frac{3 - \cos(\theta)}{2}$$

$$\leq \left\{ \frac{(1 + \cos(\theta)) + (3 - \cos(\theta))}{4} \right\}^2 = 1.$$

It reaches its extreme values at $\theta = 0$ and $\theta = \pi$, where $|S(\theta)| = 1$ and $|S(\theta)| = 0$, respectively. This implies that the modulus of $S(w)$ is never greater than one on the unit circle centered at $w = -1$ and thus $|S(z)| \leq 1$ on the left half complex plane around the negative real axis. In other words, the scheme (3.23) is A-stable and hence L-stable with either of the parameters $\lambda = 1 \pm \sqrt{2}/2$. In the implementation, we choose

$$\lambda = 1 - \frac{\sqrt{2}}{2}$$

since the local truncation error (3.26) is smaller with this parameter.

As stated, the second-order SDIRK scheme belongs to a large class of singly diagonally implicit Runge-Kutta methods. Extensions of this scheme to higher order can be found in [89].

**Remark 3.2:** For stiff ordinary differential equations, it is also common to use implicit multi-step methods, such as backward differentiation formulas (BDF)
However, we find that it is difficult to use them with adaptive mesh refinement because multi-step algorithms require initial data from several previous time steps, which may be not easily available due to the operator splitting.

3.3.4 Error Estimation and Step Size Control

In the AMR algorithm, the time step size is restricted by the condition that waves must be prevented from propagating more than one cell per time step (section 4.4.2). Because of the cost of solving linear systems, diffusion solver should not take sub-timesteps and it should be unconditionally stable. However, since the nonlinear reactions after operator splitting are spatially independent, multiple small time steps can be used to integrate them if necessary to satisfy their stability and accuracy conditions.

An adaptive time integration procedure for ordinary differential equations needs two more parts: error estimation and step size control.

To estimate errors, we can compare two sets of solutions from methods of different orders, or from the same method with different step sizes. Even we can use a pair of methods with a lower-order one embedded in a higher-order scheme [48, 88, 104].

As long as an error estimate is available, the step size is dynamically adjusted so that more accurate solution can be obtained. If the estimated error is too big, the step size is reduced so that the solution is bounded within a specified tolerance; if the error is too small, one can increase the step size to reduce the computation effort.

In this section, we will describe a Richardson extrapolation based error estimation and a simple step size control technique. For more complicated step size control strategies such as those employing a discrete PI (proportional integral) controller, refer to [80, 85, 166].
Note that a similar estimation will appear again in section 4.6 of the chapter on adaptive mesh refinement.

Suppose we first numerically integrate the following ordinary differential equation

$$\frac{d w(t)}{dt} = R(w(t)) \quad (3.31)$$

with an initial value \(w(t^n) = u^n\) by a single time step \(\Delta t\) from \(t^n\) to \(t^{n+1} = t^n + \Delta t\). Denote by \(w_1^{(0)}\) the numerical solution at the new time \(t^{n+1}\). The solution \(w_1^{(0)}\) is an approximation of the local exact solution \(w(t^{n+1})\). Let

$$e_1^{(0)} \equiv w(t^n) - w_1^{(0)}$$

be the approximation error. It is equal to the local truncation error

$$\epsilon_{\Delta t} = C_{p+1} \cdot (\Delta t)^{p+1} + C_{p+2} \cdot (\Delta t)^{p+2} + C_{p+3} \cdot (\Delta t)^{p+3} + \cdots, \quad (3.32)$$

where \(p > 0\) is the global order of the numerical scheme used. When the time step \(\Delta t\) is small enough, the coefficients \(C_{p+1+i} (i \geq 0)\) in the local truncation error (3.32) are asymptotically constant. We assume they are independent of the time step \(\Delta t\) in the rest.

Next, we halve the time step \(\Delta t\) and integrate the ODE (3.31) by two small steps to the same time \(t^{n+1}\) as above. Denote by \(w_2^{(0)}\) the new solution, which is also an approximation of the local exact solution \(w(t^{n+1})\). Let

$$e_2^{(0)} \equiv w(t^n) - w_2^{(0)}$$

be the approximation error. Note that, in each of the two small time steps, there is a local truncation error given by

$$\epsilon_{\Delta t/2} = C_{p+1} \cdot (\Delta t/2)^{p+1} + C_{p+2} \cdot (\Delta t/2)^{p+2} + C_{p+3} \cdot (\Delta t/2)^{p+3} + \cdots. \quad (3.33)$$
The approximation error $e_2^{(0)}$ equals the accumulated error, which is a double of the local truncation error $\epsilon_{\Delta t/2}$, over the time duration from $t^n$ to $t^{n+1}$.

By the arguments above, we can approximate the local exact solution in two ways:

\[
\begin{align*}
  w(t^{n+1}) &= w_1^{(0)} + e_1^{(0)} \\
  &= w_1^{(0)} + C_{p+1} \cdot (\Delta t)^{p+1} + C_{p+2} \cdot (\Delta t)^{p+2} + \cdots, \\
  w(t^{n+1}) &= w_2^{(0)} + e_2^{(0)} \\
  &= w_2^{(0)} + 2C_{p+1} \cdot (\Delta t/2)^{p+1} + 2C_{p+2} \cdot (\Delta t/2)^{p+2} + \cdots.
\end{align*}
\] (3.34a) (3.34b)

Subtracting (3.34b) from (3.34a), we get the leading order terms of the approximation errors

\[
C_{p+1} \cdot (\Delta t)^{p+1} = \frac{w_2^{(0)} - w_1^{(0)}}{1 - 2^{-p}} - C_{p+2} \frac{2^{p+1} - 1}{2^{p+1} - 2} (\Delta t)^{p+2} + \cdots,
\]

and

\[
2C_{p+1} \cdot (\Delta t/2)^{p+1} = \frac{w_2^{(0)} - w_1^{(0)}}{2^p - 1} - 4C_{p+2} \frac{2^{p+1} - 1}{2^{p+1} - 2} (\Delta t/2)^{p+2} + \cdots.
\] (3.35)

The first term in (3.35) indicates how big the approximation error $e_2^{(0)}$ is, i.e.,

\[
 e_2^{(0)} \equiv w(t^{n+1}) - w_2^{(0)} \\
 \approx 2C_{p+1} \cdot \left(\frac{\Delta t}{2}\right)^{p+1} \\
 \approx \frac{w_2^{(0)} - w_1^{(0)}}{2^p - 1}.
\]

We may determine the accuracy of the approximate solution $w_2^{(0)}$ by checking whether the absolute error satisfies

\[
|e_2^{(0)}| = \frac{|w_2^{(0)} - w_1^{(0)}|}{2^p - 1} < atol,
\]
or checking whether the relative error satisfies
\[
\frac{|e_2^{(0)}|}{|w_2^{(0)}|} = \frac{|w_2^{(0)} - w_1^{(0)}|}{(2^p - 1) \max\{|w_2^{(0)}|, \sqrt{\epsilon_{mach}}\}} < rtol.
\]

where \(\epsilon_{mach}\) is the machine roundoff; \(atol\) and \(rtol\) are tolerance for the errors.

On the other hand, we can correct the approximate solution \(w_2^{(0)}\) with the leading order term of the approximation error \(e_2^{(0)}\). We let

\[
w_2^{(1)} \equiv w_2^{(0)} + \frac{w_2^{(0)} - w_1^{(0)}}{2^p - 1} = \frac{2^p w_2^{(0)} - w_1^{(0)}}{2^p - 1}.
\]

(3.37)

It is straightforward to see that

\[
w(t^{n+1}) = w_2^{(1)} - C_{p+2} \frac{2^{p+1}}{2^p - 1} (\Delta t / 2)^{p+2} + \ldots
\]

and

\[
e_2^{(1)} \equiv w(t^{n+1}) - w_2^{(1)} = -C_{p+2} \frac{2^{p+1}}{2^p - 1} (\Delta t / 2)^{p+2} + \ldots.
\]

Here, the approximation \(w_2^{(1)}\) has one order higher than \(w_2^{(0)}\) in accuracy. Naturally it can be used to replace the numerical approximation \(u^{n+1}\).

As a matter of fact, we can further halve the time step and integrate the ODE (3.31) by four time steps of equal size \(\Delta t / 4\) again from \(t^n\) to \(t^{n+1}\). Denote by \(w_4^{(0)}\) the approximate solution. Similarly, another higher order solution \(w_4^{(1)}\) can be extrapolated by

\[
w_4^{(1)} \equiv w_4^{(0)} + \frac{w_4^{(0)} - w_2^{(0)}}{2^p - 1} = \frac{2^p w_4^{(0)} - w_2^{(0)}}{2^p - 1}.
\]

(3.38)

It has an accuracy order \(p + 2\).

Like the Romberg integration, a higher order approximate solution can even be obtained by extrapolating the two new approximations \(w_2^{(1)}\) in (3.37) and \(w_4^{(1)}\) in
(3.38) as follows

\[ w_4^{(2)} \equiv w_4^{(1)} + \frac{w_4^{(1)} - w_2^{(1)}}{2^{p+1} - 1} = \frac{2^{p+1} w_4^{(1)} - w_2^{(1)}}{2^{p+1} - 1}. \]

This approximation \( w_4^{(2)} \) has a local error of order \( p + 3 \). The absolute error \( e_4^{(1)} \) can be estimated by

\[ e_4^{(1)} \equiv w(t^{n+1}) - w_4^{(1)} \approx \frac{w_4^{(1)} - w_2^{(1)}}{2^{p+1} - 1}. \]

The procedure above can be repeated by the extrapolation formula

\[ w_{2k+1}^{(i+1)} = \frac{2^{p+i}w_{2k+1}^{(i)} - w_{2k}^{(i)}}{2^{p+i} - 1} \]

for \( i = 0, 1, \cdots \), and \( k = 0, 1, \cdots \) (see Figure 3.1), until the estimated absolute error satisfies

\[ |e_{2k+1}^{(i)}| \approx \frac{|w_{2k+1}^{(i)} - w_{2k}^{(i)}|}{2^{p+i} - 1} < atol, \]
or the relative error satisfies

\[
\frac{|e_{2k+1}^{(i)}|}{|w_{2k}^{(i)}|} \approx \frac{|w_{2k+1}^{(i)} - w_{2k}^{(i)}|}{(2p+i-1) \max\{|w_{2k+1}^{(i)}|, \sqrt{\epsilon_{\text{mach}}}\}} < rtol. \quad (3.39)
\]

In this sense, the time integration using the Richardson based error estimation is adaptive.

However, as the depth of the extrapolation increases, the computation may become very expensive. Sometimes, one should not use such a Romberg-like extrapolation procedure by halving time steps too many times.

One way of avoiding halving time steps too many times is to adjust or reduce the step sizes automatically, based on the estimated errors. This involves the technique called adaptive step size control.

Our step size control strategy is still based on the assumption that all of the coefficients \(C_{p+1+i} (i \geq 0)\) in the local truncation error (3.32) are constant or varying very slowly, when \(\Delta t\) is small enough.

Suppose the original time step \(\Delta t\) is divided into \(m\) small time steps of equal size \(\Delta t/m\) and the ODE (3.31) is integrated by \(m\) such small time steps to the point \(\Delta t\).

By our assumption, the accumulated error is approximately an \(m\) multiple of the leading order term of the local truncation error in each small time step, i.e.,

\[
e_m \equiv w(\Delta t) - w_m \approx m \cdot C \cdot \left(\frac{\Delta t}{m}\right)^{p+1}.
\]

As before, we further halve the small time steps and integrate the ODE (3.31) by \(2m\) time steps of size \(\Delta t/(2m)\). Then, we can get another approximate solution with an accumulated error

\[
e_{2m} \equiv w(\Delta t) - w_{2m} \approx 2m \cdot C \cdot \left(\frac{\Delta t}{2m}\right)^{p+1}.
\]
We let
\[ r_{2m} = 2m \cdot C \cdot \left( \frac{\Delta t}{2m} \right)^{p+1}. \] (3.40)

By the Richardson extrapolation described above, the error \( e_{2m} \) or its approximation \( r_{2m} \) can be estimated by
\[ r_{2m} \approx \frac{w_{2m} - w_m}{2^{p} - 1}. \]

Assume the absolute value of the estimated error \( r_{2m} \) is greater than the absolute tolerance \( atol \), i.e.,
\[ |r_{2m}| = 2m \cdot |C| \cdot \left( \frac{\Delta t}{2m} \right)^{p+1} > atol. \] (3.41)

Otherwise, the integration has already yielded accurate enough results.

Now, we would like to compute the number \( m^* \) of smaller time steps such that the estimated error
\[ r_{2m^*} = 2m^* \cdot C \cdot \left( \frac{\Delta t}{2m^*} \right)^{p+1} \] (3.42)
from the next Richardson extrapolation has an absolute value less than the tolerance \( atol \). In fact, from (3.40), we can get
\[ 2 \cdot C \cdot \left( \frac{\Delta t}{2} \right)^{p+1} = r_{2m} m^p. \]

Substituting the quantity above into (3.42) yields
\[ r_{2m^*} = r_{2m} \cdot \left( \frac{m}{m^*} \right)^p, \]
which is supposed to be less than \( atol \). Hence, the number \( m^* \) of smaller time steps should satisfies
\[ r_{2m} \cdot \left( \frac{m}{m^*} \right)^p < atol, \]
i.e.,
\[ m^* > m \cdot \left( \frac{r_{2m}}{atol} \right)^{1/p}. \]
In the implementation, the time step number $m^*$ is selected as the smallest integer greater than $fac \cdot m \cdot (r^m/\text{atol})^{1/p}$. That is,

$$m^* = \lfloor fac \cdot m \cdot (r^m/\text{atol})^{1/p} \rfloor.$$ 

Here, the safety factor $fac$ is introduced to reduce the number of rejected steps. It can be chosen to be a constant ($fac = 1.2$), but a better approach is probably to let it depend on the average or the maximum number of Newton iterations in the previous time steps. Once the new time step number $m^*$ is computed, we can adjust the time step size to be $\Delta t/m^*$ and continue the Richardson extrapolation process.

### 3.4 Finite Element Approximation for Linear Diffusion

Note that the transmembrane potential $V_m$ in the bidomain model is defined by $V_m = \Phi_i - \Phi_e$. The linear diffusion part (3.5) or (3.9), resulting from operator splitting applied to the bidomain model, can be rewritten as

$$\beta C_m \frac{\partial \Phi_i}{\partial t} - \beta C_m \frac{\partial \Phi_e}{\partial t} = \nabla \cdot D_i \nabla \Phi_i \quad \text{for } t^n < t < t^{n+1}, \quad (3.43a)$$

$$-\beta C_m \frac{\partial \Phi_i}{\partial t} + \beta C_m \frac{\partial \Phi_e}{\partial t} = \nabla \cdot D_e \nabla \Phi_e \quad \text{for } t^n < t < t^{n+1}. \quad (3.43b)$$

Alternatively, we can write the linear diffusion in matrix form:

$$C \frac{\partial \Phi}{\partial t} = \nabla \cdot D \nabla \Phi \quad \text{for } t^n < t < t^{n+1}, \quad (3.44)$$

where $D$ is the diffusion tensor, $C$ is the capacitance matrix and $\Phi$ is the electrical potential vector:

$$D = \begin{pmatrix} D_i \\ D_e \end{pmatrix}, \quad C = \begin{pmatrix} \beta C_m & -\beta C_m \\ -\beta C_m & \beta C_m \end{pmatrix}, \quad \Phi = \begin{pmatrix} \Phi_i \\ \Phi_e \end{pmatrix}. \quad 67$$
It is obvious to find that the linear diffusion (3.7) from the monodomain model problem has the same form as (3.44) except that the capacitance matrix $C$ is replaced by a positive scalar $\beta C_m$ and the potential vector $\Phi$ reduces to the transmembrane potential $V_m$.

In general, we are interested in solving the following initial boundary value problem (linear diffusion):

$$C \frac{\partial \Phi(t,x)}{\partial t} = \nabla \cdot D \nabla \Phi(t,x) \quad \text{for } x \in \Omega, \ t^n < t < t^{n+1}, \quad (3.45a)$$

$$n \cdot D \nabla \Phi(t,x) = 0 \quad \text{for } x \in \partial \Omega, \ t^n < t < t^{n+1}, \quad (3.45b)$$

$$\Phi(t^n, x) = \Phi^n(x) \quad \text{for } x \in \Omega, \quad (3.45c)$$

where the problem domain $\Omega$ is a polygon or a domain with Lipstchitz continuous boundary $\partial \Omega$, $n$ is the exterior normal to the domain boundary; $C$ is a symmetric and nonnegative definite matrix and $D$ is a symmetric positive definite tensor; $\Phi \in \mathbb{R}^s$ ($s = 1$ or $s = 2$) is a solution vector. However, for the sake of simplicity and to explain ideas more clearly, from now on we assume that both the unknown vector $\Phi(t,x)$ and the capacitance matrix $C$ are scalar-valued ($s = 1$) in the linear diffusion equations (3.45) unless stated explicitly.

The diffusion equations will be first discretized in space via a second-order finite element method, then in time by the Crank-Nicolson scheme, which also has second-order accuracy. The finite element method could be used for spatial discretization of the linear diffusion, in spite of the spatial heterogeneity and geometric complexity of the domain such as a heart.

On a grid without local refinement, we will use continuous piecewise-linear functions for spatial discretization. On a composite grid from local refinement where hanging nodes are present (see Figure 3.2), Lagrange multipliers are applied to enforce the continuity of data across coarse-fine subdomain interfaces [4, 21, 90, 156].
Suppose that the (polygonal) domain $\Omega$ is partitioned into quadrilateral (in two space dimensions) or hexahedral (in three space dimensions) elements. As usual, we always assume that the partition is shape-regular and conformal, or called admissible [34, 57]. Denote the partition by $T_h = \{T^{(1)}, T^{(2)}, \cdots, T^{(M)}\}$, where $M > 0$ is the number of elements in the partition and the mesh parameter $h$ is defined as the maximum of the element diameters

$$h = \max_{1 \leq i \leq M} \text{diameter}\{T^{(i)}\}.$$ 

In two space dimensions, we take the unit square as a reference element $\hat{T} = \{\xi = (\xi_1, \xi_2)^T : 0 \leq \xi_1 \leq 1, 0 \leq \xi_2 \leq 1 \}$. Define the four reference basis functions by

$$\hat{N}^{(1)}(\xi) = (1 - \xi_1)(1 - \xi_2),$$
$$\hat{N}^{(2)}(\xi) = \xi_1(1 - \xi_2),$$
$$\hat{N}^{(3)}(\xi) = (1 - \xi_1)\xi_2,$$
$$\hat{N}^{(4)}(\xi) = \xi_1\xi_2,$$

where $\xi$ is the reference coordinate. It is easy to find that, for every quadrilateral
Figure 3.3: 2D bijective mapping $J: \hat{T} \rightarrow T$
element $T$ in the partition $\mathcal{T}_h$, there is a bijective mapping $\mathbf{J}: \hat{T} \rightarrow T$, such that

$$
\mathbf{x} = \mathbf{J}(\xi) = \sum_{i=1}^{4} \mathbf{x}^{(i)} \hat{N}^{(i)}(\xi),
$$

(3.47)

where $\mathbf{x}^{(i)} = (x_1^{(i)}, x_2^{(i)})^T$ ($i = 1, \cdots, 4$) are the four vertices of the quadrilateral element $T$; they are numbered so that the orientation is preserved by the mapping as shown in Figure (3.3). With the bijective mapping $\mathbf{J}(\xi)$, a set of shape functions may be defined as $N^{(i)}(\mathbf{x}) = \hat{N}^{(i)}(\mathbf{J}^{-1}(\mathbf{x}))$ ($i = 1, \cdots, 4$), where $\mathbf{J}^{-1}(\mathbf{x})$ is the inverse of the bijective mapping $\mathbf{J}(\xi)$.

In three space dimensions, the unit cube is chosen as the reference element $\hat{T} = \{\xi = (\xi_1, \xi_2, \xi_3)^T : 0 \leq \xi_1 \leq 1, 0 \leq \xi_2 \leq 1, 0 \leq \xi_3 \leq 1 \}$, and the eight reference basis
functions are given by

\[
\hat{N}^{(1)}(\xi) = (1 - \xi_1)(1 - \xi_2)(1 - \xi_3),
\]

\[
\hat{N}^{(2)}(\xi) = \xi_1(1 - \xi_2)(1 - \xi_3),
\]

\[
\hat{N}^{(3)}(\xi) = (1 - \xi_1)\xi_2(1 - \xi_3),
\]

\[
\hat{N}^{(4)}(\xi) = \xi_1\xi_2(1 - \xi_3),
\]

\[
\hat{N}^{(5)}(\xi) = (1 - \xi_1)(1 - \xi_2)\xi_3,
\]

\[
\hat{N}^{(6)}(\xi) = \xi_1(1 - \xi_2)\xi_3,
\]

\[
\hat{N}^{(7)}(\xi) = (1 - \xi_1)\xi_2\xi_3,
\]

\[
\hat{N}^{(8)}(\xi) = \xi_1\xi_2\xi_3.
\]

For each hexahedral element \( T \) in the partition \( \mathcal{T}_h \), if we denote its eight vertices by \( \mathbf{x}^{(i)} = (x^{(i)}_1, x^{(i)}_2, x^{(i)}_3)^T \) \((i = 1, \cdots, 8)\), then the bijective mapping \( \mathbf{J}: \hat{T} \rightarrow T \) (see Figure (3.4)) can be defined similarly (the vertices are numbered such that the orientation is preserved by the mapping) as

\[
\mathbf{x} = \mathbf{J}(\xi) = \sum_{i=1}^{8} \mathbf{x}^{(i)} \hat{N}^{(i)}(\xi),
\]

and the basis functions in \( \mathbf{x} \) are \( N^{(i)}(\mathbf{x}) = \hat{N}^{(i)}(\mathbf{J}^{-1}(\mathbf{x})) \) \((i = 1, \cdots, 8)\), which are trilinear in \( \mathbf{x} \) and have the following property

\[
N^{(i)}(\mathbf{x}^{(j)}) = \delta_{ij} = \begin{cases} 
1 & \text{if } i = j, \\
0 & \text{otherwise.}
\end{cases}
\]

**Remark 3.3 :** Generally the computational domain \( \Omega \), such as the heart, may have a curve boundary \( \partial \Omega \). In this case, it is desirable to introduce more and higher order reference basis functions \( \hat{N}^{(i)}(\xi) \) for boundary elements to better approximate the boundary curve and boundary conditions (see Figure 3.3(b) and 3.4(b)). We
choose to use iso-parametric elements, where the number of basis functions used to
define the bijective mapping (3.47) or (3.49) is the same as that used to define nodal
basis functions [57]. However, in current bio-medical engineering (BME) simula-
tions, piecewise constant functions are used to represent heart geometry. The use of
piecewise linear functions here is clearly an improvement.

3.4.2 Discretization on Uniformly Refined Grids

Before discretizing the linear diffusion with finite elements, let us introduce its vari-
tional form first. Denote by $H^1(\Omega)$ the Sobolev space consisting of all functions whose
first order derivatives (in distribution sense) are all square integrable on the domain
$\Omega$. Denote by $C([t^n, t^{n+1}]; H^1(\Omega))$ the space of measurable functions from $[t^n, t^{n+1}]$
into $H^1(\Omega)$ such that, for any $\Psi \in C([t^n, t^{n+1}]; H^1(\Omega))$, $\Psi$ is continuous with re-
spect to $t$. Denote by $C^1([t^n, t^{n+1}]; L^2(\Omega))$ the space of measurable functions from
$[t^n, t^{n+1}]$ into $L^2(\Omega)$ such that, for any $\Psi \in C^1([t^n, t^{n+1}]; L^2(\Omega))$, $\Psi$ is continuously
differentiable with respect to $t$.

Multiplying the diffusion equations (3.45a) by a test function $\Psi \in H^1(\Omega)$ and
using the divergence theorem together with the homogeneous Neumann boundary
condition (3.45b), we can obtain the following variational problem:

Find $\Phi(t, x) \in C([t^n, t^{n+1}]; H^1(\Omega)) \cap C^1([t^n, t^{n+1}]; L^2(\Omega))$, such that for any
$\Psi \equiv \Psi(x) \in H^1(\Omega)$ and $t \in (t^n, t^{n+1})$,

$$\int_{\Omega} \Psi^T \cdot C \frac{\partial \Phi}{\partial t} \, dx = - \int_{\Omega} (\nabla \Psi)^T \cdot D \nabla \Phi \, dx,$$

(3.50a)

$$\int_{\Omega} \Psi^T \cdot \Phi(t^n, x) \, dx = \int_{\Omega} \Psi^T \cdot \Phi^n(x) \, dx,$$

(3.50b)

where the test function $\Psi$ is independent of time.
The variational form above is equivalent to the boundary value problem (3.45). By the assumptions on the capacitance matrix $C$ and the diffusion tensor $D$, we know that there exists a unique solution $\Phi(t, x)$, up to a constant if $C$ is a singular non-negative matrix such as in the case of the bidomain model, to the variational problem (3.50) [176, 181].

If we approximate the time derivative $\frac{\partial \Phi}{\partial t}$ by the difference
\[
\frac{\partial \Phi}{\partial t} \approx \frac{\Phi^{n+1} - \Phi^n}{\Delta t},
\]
where the time step $\Delta t = t^{n+1} - t^n$, and discretize the right hand side of the equation (3.50a) at the average of the states $\Phi(t^n, x)$ and $\Phi(t^{n+1}, x)$, we obtain the Crank-Nicolson variational form for the solution $\Phi(t^{n+1}, x)$ at new time $t = t^{n+1}$:
\[
\int_{\Omega} \Psi^T \cdot C \Phi(t^{n+1}, x) - \Phi(t^n, x) \Delta t d\Omega + \frac{\tau}{2} \int_{\Omega} (\nabla \Psi)^T \cdot D \nabla \Phi(t^{n+1}, x) d\Omega + \frac{\tau}{2} \int_{\Omega} (\nabla \Psi)^T \cdot D \nabla \Phi(t^n, x) d\Omega = b, \quad (3.52)
\]
Here, $\Phi(t^n, x) \in H^1(\Omega)$, defined by (3.50b), is the $L^2$ projection on $H^1(\Omega)$ of the initial value $\Phi^n$. It has local second-order accuracy in time. It is remarkable that the Crank-Nicolson finite element discretization for the linear diffusion falls into the category of methods of lines. It uses finite elements in space and trapezoidal rule in time. The time integration is A-stable but not L-stable. High frequency may oscillate and decay slowly. Moreover, the variational form (3.50) above can be approximated by:

Find $\Phi(t^{n+1}, x) \in H^1(\Omega)$, such that, for any $\Psi \in H^1(\Omega)$,
\[
\int_{\Omega} \Psi^T \cdot C \Phi(t^{n+1}, x) d\Omega + \tau \int_{\Omega} (\nabla \Psi)^T \cdot D \nabla \Phi(t^{n+1}, x) d\Omega = b, \quad (3.52)
\]
where the temporal parameter $\tau = \Delta t/2$ and the vector $b$ is given by

$$b = \int_{\Omega} \Psi^T \cdot C\Phi(t^n, x) \, dx - \tau \int_{\Omega} (\nabla \Psi)^T \cdot D \nabla \Phi(t^n, x) \, dx,$$

and $\Phi(t^n, x)$ is the $L^2$-projection of the initial data $\Phi^n(x)$ as defined by (3.50b).

**Remark 3.4 :** If the backward Euler scheme is used for temporal integration of the variational form (3.50) instead of the Crank-Nicolson scheme, the vector $b$ in (3.52) is replaced by $b = \int_{\Omega} \Psi^T \cdot C\Phi(t^n, x) \, dx$ and the temporal parameter becomes $\tau = \Delta t$.

**Remark 3.5 :** Note that the left hand side of the semi-discrete variational form (3.52) is a symmetric positive (non-negative in the case of the bidomain model) definite bilinear function defined on $H^1(\Omega) \times H^1(\Omega)$ and the right hand side is a continuous linear functional defined on $H^1(\Omega)$. Let $a(\Phi, \Psi)$ be the bilinear function and $f(\Psi)$ be the linear functional. Then the semi-discrete variational form (3.52) can be re-written as:

Find $\Phi^{n+1} \equiv \Phi(t^{n+1}, x) \in H^1(\Omega)$, such that

$$a(\Phi^{n+1}, \Psi) = f(\Psi) \quad \forall \Psi \in H^1(\Omega). \quad (3.53)$$

Suppose the domain $\Omega$ is uniformly (or quasi-uniformly) refined into quadrilateral (2D) or hexahedral (3D) elements and denote the partition by $T_h$ as introduced in the previous subsection. Then a finite element space of continuous piecewise bilinear (2D) or trilinear (3D) functions on the partition $T_h$ can be defined by

$$V_h = \{ \chi \in C(\bar{\Omega}) : \chi|_T \cdot J(\xi) \in Q_1(\hat{T}), \ \forall T \in T_h \},$$

where $Q_1(\hat{T})$ is the space of the bi- or trilinear functions on the reference element $\hat{T}$. It is obvious that the finite element space $V_h$ is a finite dimensional subspace of the
Sobolev space $H^1(\Omega)$. The dimension of the space equals the number of nodes in the partition $T_h$. It is well-known that on a quasi-uniform partition $T_h$, $V_h$ approximates functions in $H^2(\Omega)$ with order $h^2$.

Let $\Phi_h^n(x) \in V_h$ be an approximation of $\Phi(t^n, x)$. The finite element approximation of the variational form (3.52) reads:

\[
\int_{\Omega} \Psi_h(x)^T \cdot C \Phi_h^{n+1}(x) \, dx + \tau \int_{\Omega} (\nabla \Psi_h(x))^T \cdot D \nabla \Phi_h^{n+1}(x) \, dx = b_h,
\]  

(3.54)

where the vector $b_h$ is given by

\[
b_h = \int_{\Omega} \Psi_h(x)^T \cdot C \Phi_h^n(x) \, dx - \tau \int_{\Omega} (\nabla \Psi_h(x))^T \cdot D \nabla \Phi_h^n(x) \, dx.
\]

Note that any function $\Phi_h(x) \in V_h$ can be expressed as a linear combination of the standard finite element basis of $V_h$. In other words, suppose that the finite element subspace $V_h$ has $N$ basis functions: $\chi^{(1)}, \chi^{(2)}, \ldots, \chi^{(N)}$. Then any function $\Phi_h(x) \in V_h$ can be written as

\[
\Phi_h(x) = \sum_{i=1}^{N} \Phi_h^{(i)} \chi^{(i)},
\]

where the coefficients $\{ \Phi_h^{(i)} \}_{i=1}^{N}$ are unknown constants. Denote by $\Phi_h$ the vector of the coefficients in the linear expansion for the function $\Phi_h(x)$.

**Remark 3.6:** We do not distinguish the vector $\Phi_h$ of coefficients (unknowns) from the corresponding function $\Phi_h(x) \in V_h$.

In terms of matrices and vectors, from the fully discrete variational form (3.54) a linear system is finally obtained:

\[
(M_h + \tau K_h) \Phi_h^{n+1} = b_h,
\]  

(3.55)
where the vector \( b_h \) is given by

\[
b_h = (M_h - \tau K_h) \Phi_h^n
\]

and \( M_h = (M_h^{(i,j)})_{N \times N} \) is the mass matrix, \( K_h = (K_h^{(i,j)})_{N \times N} \) is the stiffness matrix with entries

\[
M_h^{(i,j)} = \int_{\Omega} (\chi^{(i)})^T \cdot C \chi^{(j)} \, dx \quad \text{and} \quad K_h^{(i,j)} = \int_{\Omega} (\nabla \chi^{(i)})^T \cdot D \nabla \chi^{(j)} \, dx.
\]

It can be shown that, given that the domain \( \Omega \) is uniformly partitioned with mesh parameter \( h \) and the solution \( \Phi^{n+1} \) is regular enough (e.g., \( \Phi^{n+1} \in H^2(\Omega) \)), the approximation error between the solutions of the semi-discrete variational form (3.53) and the fully discrete linear system (3.55) is bounded by multiple of the square of the mesh parameter \( h \) in the \( L^2(\Omega) \) norm, provided that the initial data is smooth enough and approximated accurately, i.e.,

\[
\| \Phi^{n+1} - \Phi_h^{n+1} \|_{L^2(\Omega)} = \| \Phi^{n+1}(\mathbf{x}) - \Phi_h^{n+1} \|_{L^2(\Omega)} \leq C h^2, \tag{3.56}
\]

where \( C \) is a constant independent of \( h \). For error estimates for parabolic equations with rough initial data, refer to [176]. On the other hand, the solution \( \Phi^{n+1} = \Phi^{n+1}(\mathbf{x}) \) approximates \( \Phi(t^{n+1}, \mathbf{x}) \) with order \( \Delta t^2 \) since the second-order Crank-Nicolson scheme is applied for time integration, i.e.,

\[
\| \Phi^{n+1}(\mathbf{x}) - \Phi(t^{n+1}, \mathbf{x}) \|_{L^2(\Omega)} < C \Delta t^2, \tag{3.57}
\]

where the constant \( C \) depends on the domain \( \Omega \) and the solution itself.

By combining the two inequalities (3.56) and (3.57), we can conclude that the Crank-Nicolson finite element scheme for the linear diffusion (3.45) has second-order accuracy both in space and time. That is, the approximation error for the linear diffusion (3.45) in a time step from \( t^n \) to \( t^{n+1} \) is given by

\[
\| \Phi_h^{n+1} - \Phi(t^{n+1}, \mathbf{x}) \|_{L^2(\Omega)} < C(h^2 + \Delta t^2). \tag{3.58}
\]
Figure 3.5: A computational domain consisting of two disjoint subdomains

The accuracy of the error depends on that with which the initial data (3.45c) can be approximated by the finite element space even though the influence of the error in the initial data fades as time increases [176].

3.4.3 Discretization on Locally Refined Grids

In the adaptive mesh refinement (AMR) algorithm, we will use a hierarchy of properly nested grids to define the computational domain $\Omega$. Our AMR algorithm always works with at most two grids in any part of the domain. Thus it will suffice locally to assume that the computational domain $\Omega$ consists of two disjoint subdomains, a coarse region $\Omega_c$ covered by a coarse grid and a fine region $\Omega_f$ defined by a fine grid (see Figure 3.5). The two subdomains $\Omega_c$ and $\Omega_f$ satisfy $\bar{\Omega} = \bar{\Omega}_c \cup \bar{\Omega}_f$ and $\Omega_c \cap \Omega_f = \emptyset$. Let $\Gamma = \partial \Omega_c \cap \partial \Omega_f$ be the smooth interface between the subdomains.
For simplicity, we assume in this subsection that the mesh parameter of the fine grid covering Ω_f is half of that of the coarse grid covering Ω_c, i.e., the refinement ratio equals two in the AMR algorithm. Let \( G_h \) be the coarse grid and \( G_{h/2} \) be the fine grid. In the meantime, the fine domain Ω_f is also denoted by Ω_{h/2} and the coarse domain is denoted by Ω_h.

Denote the restrictions to the subdomains Ω_c and Ω_f of the solution vector \( \Phi \) in the initial boundary value problem (3.45) by \( \Phi_f \) and \( \Phi_c \), respectively. The solution vectors \( \Phi_c \) and \( \Phi_f \) must satisfy the following continuity conditions along the coarse-fine subdomain interface \( \Gamma \):

\[
\Phi_f = \Phi_c \quad \text{and} \quad n_f \cdot D \nabla \Phi_f + n_c \cdot D \nabla \Phi_c = 0, \quad (3.59)
\]

where \( n_f \) and \( n_c \) are the exterior unit normals to the interface \( \Gamma \) with respect to the subdomains Ω_f and Ω_c, respectively; actually, the two normals \( n_f \) and \( n_c \) have opposite signs along the interface \( \Gamma \).

It is not difficult to find that the corresponding variational form of the initial boundary value problem in the subdomains are given by:

Find \( \Phi_c(t, x) \in C([t^n, t^{n+1}]; H^1(\Omega_c)) \cap C^1([t^n, t^{n+1}]; L^2(\Omega_c)) \) and \( \Phi_f(t, x) \in C([t^n, t^{n+1}]; H^1(\Omega_f)) \cap C^1([t^n, t^{n+1}]; L^2(\Omega_f)) \), such that

\[
\int_{\Omega_c} \Psi^T_c \cdot C \frac{\partial \Phi_c}{\partial t} \, dx = - \int_{\Omega_c} (\nabla \Psi_c)^T \cdot D \nabla \Phi_c \, dx \]
\[
+ \int_{\Gamma} \Psi^T_c \cdot [n_c \cdot (D \nabla \Phi_c)] \, ds, \quad (3.60a)
\]

\[
\int_{\Omega_f} \Psi^T_f \cdot C \frac{\partial \Phi_f}{\partial t} \, dx = - \int_{\Omega_f} (\nabla \Psi_f)^T \cdot D \nabla \Phi_f \, dx \]
\[
+ \int_{\Gamma} \Psi^T_f \cdot [n_f \cdot (D \nabla \Phi_f)] \, ds, \quad (3.60b)
\]

79
for \( t \in (t^n, t^{n+1}) \), \( \forall \Psi_c \equiv \Psi_c(x) \in H^1(\Omega_c) \) and \( \forall \Psi_f \equiv \Psi_f(x) \in H^1(\Omega_f) \). In addition, the Steklov-Poincare interface continuity conditions (3.59) are enforced in a weak sense, for \( t \in [t^n, t^{n+1}] \),

\[
\int_{\Gamma} \lambda_f^T \cdot \Phi_f(t, x) \, ds = \int_{\Gamma} \lambda_c^T \cdot \Phi_c(t, x) \, ds, \quad \forall \lambda_f \in H^{-\frac{1}{2}}(\Gamma), \quad (3.61a)
\]

\[
\int_{\Gamma} \mu_c^T \cdot [n_c \cdot (D\nabla \Phi_c(t, x))] \, ds = -\int_{\Gamma} \mu_f^T \cdot [n_f \cdot (D\nabla \Phi_f(t, x))] \, ds \quad \forall \mu_c \in H^{\frac{1}{2}}(\Gamma). \quad (3.61b)
\]

The variational problem (3.60)-(3.61) has a unique solution, up to a constant if the capacitance matrix \( C \) is singular in the case of the bidomain model.

Furthermore, similar to the semi-discrete problem (3.52), by the Crank-Nicolson scheme or the backward Euler scheme, we can get the following equivalent semi-discrete problem:

Find \( \Phi_c(t^{n+1}, x) \in H^1(\Omega_c) \) and \( \Phi_f(t^{n+1}, x) \in H^1(\Omega_f) \), such that

\[
\int_{\Omega_f} \Psi_f^T \cdot C\Phi_f(t^{n+1}, x) \, dx + \tau \int_{\Omega_f} (\nabla \Psi_f)^T \cdot D\nabla \Phi_f(t^{n+1}, x) \, dx \\
- \tau \int_{\Gamma} \Psi_f^T \cdot [n_f \cdot (D\nabla \Phi_f(t^{n+1}, x))] \, ds = b_f \quad \forall \Psi_f \in H^1(\Omega_f), \quad (3.62a)
\]

\[
\int_{\Omega_c} \Psi_c^T \cdot C\Phi_c(t^{n+1}, x) \, dx + \tau \int_{\Omega_c} (\nabla \Psi_c)^T \cdot D\nabla \Phi_c(t^{n+1}, x) \, dx \\
- \tau \int_{\Gamma} \Psi_c^T \cdot [n_c \cdot (D\nabla \Phi_c(t^{n+1}, x))] \, ds = b_c \quad \forall \Psi_c \in H^1(\Omega_c), \quad (3.62b)
\]

with the data \( \Phi_f(t^{n+1}, x) \) and \( \Phi_c(t^{n+1}, x) \) at new time \( t = t^{n+1} \) satisfying the Steklov-
Poincare weak continuity conditions

\[ \int_{\Gamma} \lambda_f^T \cdot \Phi_f(t^{n+1}, x) \, ds = \int_{\Gamma} \lambda_c^T \cdot \Phi_c(t^{n+1}, x) \, ds, \]

\( \forall \lambda_f \in H^{-\frac{1}{2}}(\Gamma), \) \hspace{1cm} (3.63a)

\[ \int_{\Gamma} \mu_c^T \cdot [n_c \cdot (D\nabla \Phi_c(t^{n+1}, x))] \, ds = - \int_{\Gamma} \mu_f^T \cdot [n_f \cdot (D\nabla \Phi_f(t^{n+1}, x))] \, ds \]

\( \forall \mu_c \in H^\frac{1}{2}(\Gamma), \) \hspace{1cm} (3.63b)

where the vector \( b_c \) and \( b_f \) are determined by the old time values \( \Phi_c(t^n, x) \) and \( \Phi_f(t^n, x) \); \( \tau \) is the full time step \( \Delta t = t^{n+1} - t^n \) or a half time step \( \Delta t/2 \), depending on the time integration scheme used.

If the first-order backward Euler scheme is used for the semi-discretization, the temporal parameter is \( \tau = \Delta t \) and the vectors \( b_f \) and \( b_c \) are defined by

\[ b_f = \int_{\Omega_f} \Psi_f^T \cdot C\Phi_f(t^n, x) \, dx, \]

\[ b_c = \int_{\Omega_c} \Psi_c^T \cdot C\Phi_c(t^n, x) \, dx. \]

If the second-order Crank-Nicolson scheme is used for the semi-discretization, the temporal parameter is \( \tau = \Delta t/2 \) and the vectors \( b_f \) and \( b_c \) are given by

\[ b_f = \int_{\Omega_f} \Psi_f^T \cdot C\Phi_f(t^n, x) \, dx - \tau \int_{\Omega_f} (\nabla \Psi_f)^T \cdot D\nabla \Phi_f(t^n, x) \, dx \]

\[ + \tau \int_{\Gamma} \Psi_f^T \cdot [n_f \cdot (D\nabla \Phi_f(t^n, x))] \, ds, \]

\[ b_c = \int_{\Omega_c} \Psi_c^T \cdot C\Phi_c(t^n, x) \, dx - \tau \int_{\Omega_c} (\nabla \Psi_c)^T \cdot D\nabla \Phi_c(t^n, x) \, dx, \]

\[ + \tau \int_{\Gamma} \Psi_c^T \cdot [n_c \cdot (D\nabla \Phi_c(t^n, x))] \, ds. \]
It can be shown that there exists a unique solution, up to a constant if the capacitance matrix $C$ is singular such as in the case of the bidomain model, to the semi-discrete variational problem (3.62)-(3.63), by using the Poincare-Friedrich’s inequality with respect to a graph norm of data on the composite domain $\bar{\Omega} = \bar{\Omega}_c \cup \bar{\Omega}_f$.

Similar to the case of uniform refinement, two finite dimensional spaces can be defined. The first one consists of continuous piecewise bi-/trilinear functions on the coarse grid $G_h$, also called the coarse partition $T_h$ of the coarse domain $\Omega_h$. It is defined by

$$V_h = \{ \chi \in C(\bar{\Omega}_h) : \chi|_T \cdot J(\xi) \in Q_1(\hat{T}), \quad \forall T \in T_h \},$$

where $Q_1(\hat{T})$ is the space of the bi- or trilinear functions on the reference element $\hat{T}$. The second one is composed of continuous piecewise bilinear or trilinear functions on the fine grid $G_{h/2}$, also called the fine partition $T_{h/2}$ of the fine domain $\Omega_{h/2}$. It is defined by

$$V_{h/2} = \{ \chi \in C(\bar{\Omega}_{h/2}) : \chi|_T \cdot J(\xi) \in Q_1(\hat{T}), \quad \forall T \in T_{h/2} \}.$$

The two finite dimensional spaces $V_h$ and $V_{h/2}$ are the subspaces of the Sobolev spaces $H^1(\Omega_h)$ and $H^1(\Omega_{h/2})$, respectively. The space dimension of $V_h$ equals the number of nodes in the coarse partition $T_h$; the space dimension of $V_{h/2}$ equals the number of nodes in the fine partition $T_{h/2}$.

Let $\Phi^i_{h/2} \in V_{h/2}$ be (the vector of nodal unknowns of) an approximation of $\Phi_f(t^n, x)$; $\Phi^i_h \in V_h$ be (the vector of nodal unknowns of) an approximation of $\Phi_c(t^n, x)$. Let $S_h$ be the space of functions that are restrictions to the interface $\Gamma$ of those in $V_h$; $S_{h/2}$ be the space of functions that are restrictions to the interface $\Gamma$ of those in $V_{h/2}$; $L_{h/2}$ is the dual space of $S_{h/2}$.

Then we find that the finite element discretization of the variational problem (3.62)-(3.63) reads:  

82
Find $\Phi_{h/2}^{n+1} \in V_{h/2}$ and $\Phi_{h}^{n+1} \in V_{h}$, such that

$$
\int_{\Omega_{h/2}} \Psi_{h/2}^{T} \cdot C\Phi_{h/2}^{n+1} \, dx + \tau \int_{\Omega_{h/2}} (\nabla \Psi_{h/2})^{T} \cdot D\nabla \Phi_{h/2}^{n+1} \, dx
$$

$$
- \tau \int_{\Gamma} \Psi_{h/2}^{T} \cdot [n_{f} \cdot (D\nabla \Phi_{h/2}^{n+1})] \, ds = b_{h/2} \quad \forall \Psi_{h/2} \in V_{h/2},
$$

(3.66a)

$$
\int_{\Omega_{h}} \Psi_{h}^{T} \cdot C\Phi_{h}^{n+1} \, dx + \tau \int_{\Omega_{h}} (\nabla \Psi_{h})^{T} \cdot D\nabla \Phi_{h}^{n+1} \, dx
$$

$$
- \tau \int_{\Gamma} \Psi_{h}^{T} \cdot [n_{c} \cdot (D\nabla \Phi_{h}^{n+1})] \, ds = b_{h} \quad \forall \Psi_{h} \in V_{h},
$$

(3.66b)

with the two solution vectors $\Phi_{h/2}^{n+1}$ and $\Phi_{h}^{n+1}$ subject to the constraints

$$
\int_{\Gamma} \lambda_{h/2}^{T} \cdot \Phi_{h/2}^{n+1} \, ds = \int_{\Gamma} \lambda_{h/2}^{T} \cdot \Phi_{h}^{n+1} \, ds,
$$

$$
\forall \lambda_{h/2} \in L_{h/2} \subset H^{-\frac{1}{2}}(\Gamma),
$$

(3.67a)

$$
\int_{\Gamma} \mu_{h}^{T} \cdot [n_{c} \cdot (D\nabla \Phi_{h}^{n+1})] \, ds = - \int_{\Gamma} \mu_{h}^{T} \cdot [n_{f} \cdot (D\nabla \Phi_{h/2}^{n+1})] \, ds
$$

$$
\forall \mu_{h} \in S_{h} \subset H^{\frac{1}{2}}(\Gamma).
$$

(3.67b)

Here, the temporal parameter $\tau$ is equal to a half time step $\Delta t/2$; the vectors $b_{h}$ and $b_{h/2}$ are respectively determined by the old time approximations $\Phi_{h}^{n}$ and $\Phi_{h/2}^{n}$ as follows

$$
b_{h/2} = \int_{\Omega_{h/2}} \Psi_{h/2}^{T} \cdot C\Phi_{h/2}^{n} \, dx - \tau \int_{\Omega_{h/2}} (\nabla \Psi_{h/2})^{T} \cdot D\nabla \Phi_{h/2}^{n} \, dx
$$

$$
+ \tau \int_{\Gamma} \Psi_{h/2}^{T} \cdot [n_{f} \cdot (D\nabla \Phi_{h/2}^{n})] \, ds,
$$

$$
b_{h} = \int_{\Omega_{h}} \Psi_{h}^{T} \cdot C\Phi_{h}^{n} \, dx - \tau \int_{\Omega_{h}} (\nabla \Psi_{h})^{T} \cdot D\nabla \Phi_{h}^{n} \, dx
$$

$$
+ \tau \int_{\Gamma} \Psi_{h}^{T} \cdot [n_{c} \cdot (D\nabla \Phi_{h}^{n})] \, ds,
$$

83
Figure 3.6: A two-level composite grid

in the case that the second-order Crank-Nicolson scheme is used for time integration.

Furthermore, by choosing the standard finite element basis for each of the finite dimensional subspaces $\mathbf{V}_h$, $\mathbf{V}_{h/2}$, $\mathbf{S}_h$ and $\mathbf{L}_{h/2}$, the integral identities (3.66) in the discrete approximation will have the following form:

\begin{align*}
(M_{h/2} + \tau K_{h/2}) \Phi_{h/2}^{n+1} - \tau C_{h/2} \Phi_{h/2}^{n+1} &= b_{h/2}, \\
(M_h + \tau K_h) \Phi_h^{n+1} + \tau C_h \Phi_h^{n+1} &= b_h,
\end{align*}

(3.69a) (3.69b)

where $M_{h/2}$, $M_h$ are the mass matrices; $K_{h/2}$ and $K_h$ are the stiffness matrices; and $C_{h/2} \Phi_{h/2}^{n+1}$ and $C_h \Phi_h^{n+1}$ are the flux through the coarse-fine subdomain interface $\Gamma$.

Since we are working with piecewise linear finite element spaces, it is natural to define the interface prolongation operator $\mathcal{P}_{h/2}^h$ by a linear interpolation $\mathcal{I}_{h/2}^h$:

$\Psi_h^{n+1} \in S_h \mapsto \mathcal{I}_{h/2}^h \Psi_h^{n+1} \in S_{h/2}$; define the interface restriction operator $\mathcal{R}_{h/2}^h$ as the adjoint of the prolongation operator, i.e., $\mathcal{R}_{h/2}^h = (\mathcal{P}_{h/2}^h)^T \equiv \mathcal{I}_{h/2}^h : \Psi_{h/2}^{n+1} \in S_{h/2} \mapsto$
\[ \mathcal{I}_{h/2}^{h} \Psi_{h}^{n+1} \in S_{h}. \]

From the discrete Steklov-Poincare constraints (3.67), it is not difficult to find that

\[ \Phi_{h/2}^{n+1} = \mathcal{P}_{h/2}^{h} \Phi_{h}^{n+1} = \mathcal{I}_{h/2}^{h} \Phi_{h}^{n+1}, \]
\[ C_{h} \Phi_{h}^{n+1} = \mathcal{R}_{h/2}^{h} C_{h/2} \Phi_{h/2}^{n+1} = \mathcal{I}_{h/2}^{h} C_{h/2} \Phi_{h/2}^{n+1}. \]

Here, the fluxes through the coarse-fine subdomain interface, i.e., the second terms in the equations (3.69), are connected by the restriction operator \( \mathcal{R}_{h/2}^{h} \equiv \mathcal{I}_{h/2}^{h} \); and the solutions \( \Phi_{h}^{n+1} \) and \( \Phi_{h/2}^{n+1} \) along the interface \( \Gamma \) are connected by the prolongation operator \( \mathcal{P}_{h/2}^{h} \equiv \mathcal{I}_{h/2}^{h} \). In other words, the data at the hanging nodes on the fine grid side are obtained by linear interpolation of those on the coarse grid side. The solution is thus continuous along the coarse fine interface, belonging to the Sobolev space \( H^{1}(\Omega) \). In this sense, the finite element discretization is conforming although the composite grid is not conformal (see Figure 3.6).

Moreover, let \( m_{h/2}^{n+1} = \tau C_{h/2} \Phi_{h/2}^{n+1} \) and \( m_{h}^{n+1} = \tau C_{h} \Phi_{h}^{n+1} \). we have the following composite grid equations:

\[ (M_{h/2} + \tau K_{h/2}) \Phi_{h/2}^{n+1} - m_{h/2}^{n+1} = b_{h/2}, \quad (3.71a) \]
\[ (M_{h} + \tau K_{h}) \Phi_{h}^{n+1} + m_{h}^{n+1} = b_{h}, \quad (3.71b) \]

subject to the constraints along the coarse-fine subdomain interface

\[ \Phi_{h/2}^{n+1} = \mathcal{P}_{h/2}^{h} \Phi_{h}^{n+1}, \quad (3.72a) \]
\[ m_{h}^{n+1} = \mathcal{R}_{h/2}^{h} m_{h/2}^{n+1}, \quad (3.72b) \]

where the interface prolongation operator \( \mathcal{P}_{h/2}^{h} \equiv \mathcal{I}_{h/2}^{h} \) and the interface restriction operator \( \mathcal{R}_{h/2}^{h} \equiv \mathcal{I}_{h/2}^{h} \).
Chapter 4

Adaptive Mesh Refinement

The adaptive mesh refinement (AMR) algorithm for the singularly perturbed reaction-diffusion systems from computational cardiology is described in detail in this chapter. First, we make a brief introduction to the AMR algorithm that we are about to propose. In section 4.2, the basic assumptions for our AMR algorithm are presented. After that, we prescribe the hierarchical grid system in section 4.3. Our AMR algorithm follows Berger-Oliger’s time stepping strategy [27, 181]. Under the framework proposed by them, the fundamental components of our AMR are described in section 4.4. The treatment of initial and boundary conditions in the AMR algorithm is discussed in section 4.5. Finally, in the last section, we propose an error estimation procedure based on the Richardson extrapolation technique.

4.1 Introduction

The adaptive mesh refinement algorithm uses a hierarchy of properly nested levels. Each level has a main grid, which consists of quadrilateral elements (cells) in two space dimensions (2D) or hexahedral elements (cells) in three space dimensions (3D). Here, elements are usually called cells in the AMR algorithm. The main grid on a fine level is generated through regular bisection or refinement from a subgrid of that on a coarse level. We represent a grid by lists of cells and their lower dimensional facets, including nodes, edges and sides. The list-based data structure together with a hierarchical ordering for grid entities allows us to perform uniform refinement easily. The refinement ratio may be chosen to be any even positive integer number, if necessary, without imposing unnecessary computer memory overheads.
The AMR algorithm follows Berger-Oliger’s approach in time stepping. The hierarchical system is advanced recursively. Starting from the coarsest level, called the root level, of the system, a coarse level is first integrated with a coarse time step, followed by integrating a fine level several times with a smaller time step until both levels are synchronized. Upon synchronization, some typical routines as in Berger-Oliger’s approach are invoked.

4.2 Basic Assumptions

Before introducing the AMR algorithm in detail, we would like to briefly list a few basic assumptions, following the lines proposed by Trangenstein [180, 181]. For more details, refer to the original papers and references therein.

Suppose that an initial coarse grid is given. It may be generated by any grid generator. The coarse grid must consist of quadrilateral cells (2D) or hexahedral cells (3D). Since the coarse grid must completely cover the entire physical domain and the domain may be geometrically complex, it is possible that the grid is unstructured, or at most block-structured. Typically, we require that the number of cells in the coarse grid is small so that linear systems on this coarse grid can be solved very quickly in the middle of a multigrid iteration. The coarse grid will be the central part of the root level of the hierarchical system. It is fixed during the process of adaptive mesh refinement. Other grids on fine levels are generated dynamically and recursively by local or uniform refinement of those on coarse levels.

First, we assume that the region covered by the main grid on a fine level is contained in the interior of that on the coarser level unless both coarse and fine grids coincides with the physical boundary of computational domain. This assumption can prevent recursive searching for cell neighbors. This assumption is called Proper Level Nesting.
Second, we assume that if a cell of a coarse grid is refined in any part of its physical space, then it is refined everywhere. As a result, the boundary of the main grid on a fine level aligns with the boundary of a subgrid of that on the previous coarser level. By a subgrid of a grid, we mean the union of a subset of cells of the grid. This assumption is called **Alignment of Level Grids**.

Third, we assume that we are given an even integer refinement ratio \( r \). Whenever a coarse cell is refined, it is subdivided into \( r^d \) sub-cells, where \( d \) is the number of space dimensions. In our current implementation of the AMR algorithm, the refinement ratio \( r \) is restricted to be an even integer number and all levels in the hierarchical system use the same refinement ratio. Nevertheless, we call the assumption as **Flexible Refinement Ratio** since, at least, the refinement ratio can be any even integer number.

Fourth, after we advance the data on a coarse level grid to some time, we assume that the data on its finer level are advanced by as many time steps as required by stability and accuracy to reach exactly the same time as the coarse level. This assumption implies that a coarse level is integrated before its finer levels and the time step on a coarse level is an integer multiple of that on the next finer level. It also implies that the time stepping algorithm must be applied recursively within each time step on all but the finest level. This assumption is called **Synchronization of Advancement**.

Fifth, by the time a fine level and its coarser level are synchronized, we assume that data on the fine level is more accurate than that on the coarse level. So, the data on a fine level must be upscaled to its coarser level before the coarse level is further advanced by another coarse time step. This assumption is called **Fine Data Preference**

As stated, all grids except the coarsest one in the AMR algorithm are changing
dynamically. It is necessary for a coarse level to regrid its finer level from time to time. But, it will be extremely costly to tag cells and regrid levels every coarse time step. So, we would rather make regridding infrequently. On a coarse level, the number of time steps between times of regridding its finer level is called regrid interval, denoted by \( \kappa \). In the AMR algorithm, the regrid interval \( \kappa \) may be chosen to be an integer divisor of the refinement ratio \( r \). This assumption is called **Infrequent Regridding**.

### 4.3 Hierarchical Grid System

The hierarchical grid system employed in our AMR algorithm is different from Berger-Oliger’s approach. It is not restricted to Cartesian or logically rectangular grids. With our representation of unstructured grids, problems on geometrically complex domains can be naturally handled like a typical finite element software (e.g., the deal.II code [11]). Another point associated with the representation is that, unlike Berger-Oliger’s strategy, there is no need to include unnecessarily refined cells to maintain union of rectangular patches during local grid refinement, and hence the locally refined regions are minimized in our approach.

#### 4.3.1 Grid Representation

In the AMR algorithm, we define a grid by lists of cells and their lower dimensional facets, including nodes, edges and sides. The list-based data structure together a hierarchical ordering of grid entities allows us to make uniform refinement easily. The hierarchical ordering will be discussed in detail in the next subsection.

In one space dimension, a grid consists of a list of edges and a list of nodes. Each node in the node list is defined by its physical coordinate and each edge in the edge list is defined by two indices to its endpoints in the node list.
In two space dimensions, a grid is represented by a list of nodes, a list of edges and a list of cells. The definitions of node and edge are the same as in one space dimension except that the coordinates of a node are a pair of real numbers. A cell in two space dimensions is actually a quadrilateral element. It is defined by four indices to its edges in the edge list and another four indices to its nodes in the node list.

In three space dimensions, a grid is defined by a list of nodes, a list of edges, a list of sides and a list of cells. A side in three space dimensions is a four edge polygon. Its definition is exactly the same as that of quadrilateral cells in two space dimensions. A cell in three space dimensions is a hexahedral element, which consists of six indices to its sides in the side list, twelve indices to its edges in the edge list and eight indices to its vertices in the node list. Similarly, the definitions of node and edge are the same as in lower dimensional space except that the coordinates of a node is a triple of real numbers.

By our representation of a grid, the connectivity of entities within a grid is clear. We can easily find out adjacent cells for every cell, every side, or every node. It is worth noting that a grid with this representation may have several disconnected components.

One advantage of the grid representation is that, in implementation, we can allocate and de-allocate memory only once for all entities (nodes, edges, sides and cells), assuming that the numbers of nodes, edges, sides and cells have been counted or known, since each entity has a fixed length and does not use any pointer. The nodes, edges and sides are stored only one time in each cell that owns them. This may involve less computer storage than storing the information cell by cell with possible duplication of shared information for neighboring cells.

In addition, we have to point out that there must be some attributes associated with each grid entity. For example, a grid entity, such as node, edge, side and cell,
needs to have a boolean flag to indicate whether it is on the boundary or interior of a grid.

Note that a codimension-one grid entity has two adjacent cells unless it is on the boundary of a grid. It is this property that we use to identify whether a codimension-one grid entity is on boundary or not. After codimension-one boundary grid entities are identified, all of the lower dimensional entities directly connected to them are also marked as “on-the-boundary”.

4.3.2 Hierarchical Ordering

To facilitate refinement of grids, data transferring among grids and even relaxation in multigrid iteration, the grid entities (cells, sides, edges and nodes) of a grid are ordered hierarchically.

In our AMR algorithm, the cells are ordered by a space filling curve (SFC), such as the Morton SFC and the Hilbert-Peano SFC. For the advantages of SFC for cell ordering, refer to [50, 112]. The grids in Figure 4.1 use a Morton SFC. The grids in Figure 4.2 use a Hilbert-Peano SFC. More space filling curves (2D) with different
level indices are shown in Figures 4.3 and 4.4.

The lower dimensional grid entities are ordered differently from the cells. In the ordering, boundary entities are ordered first and interior entities are ordered next. See Figures 4.5 and 4.6 for examples of the hierarchical ordering. The details of the ordering for lower dimensional grid entities will be discussed in the next subsection on uniform grid refinement.

4.3.3 Uniform Grid Refinement

Suppose that we are given a coarse grid, which has been ordered such that boundary entities are indexed first and interior entities are indexed next. The grid is to be uniformly refined to generate a fine grid. For simplicity and conciseness, here we only consider the case that the refinement ratio is $r = 2$. In the next, we call the grid entities in the coarse grid as the old entities and those in the fine grid as the new entities.

In one space dimension, a cell is simply a line segment. Subdividing a cell will generate one new node and two new cells, called sub-cells. There are just a few
Figure 4.3: Morton space filling curves
Figure 4.4: Hilbert-Peano space filling curves
boundary nodes in one space dimension. Here, note that the boundary may have more than two nodes since a grid in the representation may have several disconnected components. A node can not be subdivided, so there are no new boundary nodes resulting from grid refinement in one space dimension. Hence, the ordering of nodes in the fine grid is the old boundary nodes first, the old interior nodes next, followed by the new interior nodes, which are centers of the old cells in the coarse grid.

In two space dimensions, a cell will generate one new node, four new edges and four new sub-cells. Each edge will generate one new node and two new edges. Note that the new node and new edges from a boundary edge are also on the boundary and the new node and new edges from an interior edge is also in the interior. The ordering of nodes in the fine grid is the old boundary nodes first, then the new boundary nodes from refinement of the old boundary edges, the old interior nodes next, followed by the new interior nodes which are centers of the old interior edges, and finally the new interior nodes which are centers of the old cells. The ordering of edges in the fine grid is the new boundary edges from refinement of the old boundary edges first, then the new interior edges from refinement of the old interior edges, followed by the new interior edges from refinement of the old interior edges, and finally the new interior edges from refinement of the old cells. For examples of uniform grid refinement in two space dimensions, see Figures 4.5 and 4.6.

In three space dimensions, a cell will give one new node, six new edges, twelve new sides and eight new sub-cells; a side will generate one new node, four new edges and four new sides; an edge will generate one new node and two new edges. The ordering of nodes is the old boundary nodes first, the new boundary nodes from refinement of the old boundary edges next, then the new boundary nodes from refinement of the old boundary sides, followed by the old interior nodes, the new interior nodes from refinement of the old interior edges, the new interior nodes from refinement of the old
(a) Ordering of nodes for coarse grid $G_h$

(b) Ordering of edges for coarse grid $G_h$

(c) Ordering of nodes for fine grid $G_{h/2}$

(d) Ordering of edges for fine grid $G_{h/2}$

(e) Ordering of nodes for finer grid $G_{h/4}$

(f) Ordering of edges for finer grid $G_{h/4}$
(g) Ordering of nodes for finest grid $G_{h/8}$

(h) Ordering of edges for finest grid $G_{h/8}$

**Figure 4.5**: Hierarchical ordering of lower dimensional grid entities (cells are ordered by a Morton space filling curve)
(a) Ordering of nodes for coarse grid $G_h$

(b) Ordering of edges for coarse grid $G_h$

(c) Ordering of nodes for fine grid $G_{h/2}$

(d) Ordering of edges for fine grid $G_{h/2}$

(e) Ordering of nodes for finer grid $G_{h/4}$

(f) Ordering of edges for finer grid $G_{h/4}$
**Figure 4.6**: Hierarchical ordering of lower dimensional grid entities (cells are ordered by a Hilbert-Peano space filling curve)
interior sides, and finally the new interior nodes corresponding to the centers of the old cells. Similarly, the ordering of edges is the new boundary edges from refinement of the old boundary edges first, the new boundary edges from refinement of the old boundary sides next, then the new interior edges from refinement of the old interior edges, followed by the new interior edges from refinement of the old interior sides, and finally the new interior edges from refinement of the old cells. The ordering of sides is the new boundary sides from refinement of the old boundary sides first, the new interior sides from refinement of the old interior sides next, followed by the new interior sides from refinement of the old cells.

In addition, the numbers of nodes, edges, sides and cells in the fine grid can be easily computed. For example, in three dimensions, suppose that the coarse grid has $N_{\text{bdry\_node}}^{(c)}$ boundary nodes, $N_{\text{bdry\_edge}}^{(c)}$ boundary edges, $N_{\text{bdry\_side}}^{(c)}$ boundary sides, $N_{\text{int\_node}}^{(c)}$ interior nodes, $N_{\text{int\_edge}}^{(c)}$ interior edges, $N_{\text{int\_side}}^{(c)}$ interior sides and $N_{\text{int\_cell}}^{(c)}$ interior cells. Then the fine grid has $N_{\text{bdry\_node}}^{(f)} = N_{\text{bdry\_node}}^{(c)} + N_{\text{bdry\_edge}}^{(c)} + N_{\text{bdry\_side}}^{(c)}$ boundary nodes, $N_{\text{bdry\_edge}}^{(f)} = 2N_{\text{bdry\_edge}}^{(c)} + 4N_{\text{bdry\_side}}^{(c)}$ boundary edges, $N_{\text{bdry\_side}}^{(f)} = 4N_{\text{bdry\_side}}^{(c)}$ boundary sides, $N_{\text{int\_node}}^{(f)} = N_{\text{int\_node}}^{(c)} + N_{\text{int\_edge}}^{(c)} + N_{\text{int\_side}}^{(c)} + N_{\text{int\_cell}}^{(c)}$ interior nodes, $N_{\text{int\_edge}}^{(f)} = 2N_{\text{int\_edge}}^{(c)} + 4N_{\text{int\_side}}^{(c)} + 6N_{\text{int\_cell}}^{(c)}$ interior edges, $N_{\text{int\_side}}^{(f)} = 4N_{\text{int\_side}}^{(c)} + 12N_{\text{int\_cell}}^{(c)}$ interior sides and $N_{\text{int\_cell}}^{(f)} = 8N_{\text{int\_cell}}^{(c)}$ interior cells. In the implementation, this makes it possible to allocate the memory only once for a uniformly refined grid. Hence, memory allocation times are greatly reduced.

With the hierarchical ordering, the grid entity correspondence can be easily inferred. There is no need to maintain lists of correspondence indices. For example, assume that a node in a fine grid has an index $I_{\text{node}}^{(f)}$ and its value is greater than $N_{\text{bdry\_node}}^{(f)}$ but less than $N_{\text{bdry\_node}}^{(f)} + N_{\text{int\_node}}^{(c)}$, then we can conclude by the hierarchical ordering that this node corresponds to a node in the coarse grid and its index in the
grid is \( I^{(c)}_{\text{node}} = I^{(f)}_{\text{node}} - N^{(f)}_{\text{bdry node}} \). If the fine index \( I^{(f)}_{\text{node}} \) is between \( N^{(f)}_{\text{bdry node}} \) and \( N^{(f)}_{\text{bdry node}} + N^{(c)}_{\text{int node}} + N^{(c)}_{\text{int edge}} \), then we can say the node corresponds to an edge in the coarse grid and the index of the edge is \( I^{(c)}_{\text{edge}} = I^{(f)}_{\text{node}} - N^{(f)}_{\text{bdry node}} - N^{(c)}_{\text{int node}} \).

That is, by checking which interval that the value of \( I^{(f)}_{\text{node}} \) lies in and computing the offset, we can find the type of the corresponding grid entity and its index in the coarse grid. As a matter of fact, by the hierarchical ordering, the grid entities from refinement of the same type of entities have been ordered and grouped together. For example, the nodes from refinement of interior edges are all ordered before those from refinement of interior sides. The edges from refinement of cells are all ordered after those from refinement of sides. During data communication between coarse fine levels, such as up-/downscaling, prolongation and restriction in a multigrid iteration, it is not necessary to make one-by-one checking and computing. The corresponding grid entity and index in the coarse grid can be easily and naturally obtained just by group-by-group looping over all nodes, edges, sides or cells.

### 4.3.4 Local Grid Refinement

Let \( G_c \) represent the main grid on a coarse level \( \ell_c \). To perform local refinement, we first use an error estimation procedure to tag those cells in \( G_c \) that have large error data and all other cells sufficiently near the ones with large error. Details of tagging of cells and error estimation will be discussed in subsection 4.4.4 and section 4.6. Next, we combine all of the tagged cells together to get a subgrid of the coarse grid \( G_c \). Denote the subgrid by \( \hat{G}_c \). Note that the grid entities in the subgrid \( \hat{G}_c \) are in general not ordered as boundary first and interior next as in the hierarchical ordering. So, a re-ordering should be additionally performed. Finally, we make a uniform refinement for the subgrid \( \hat{G}_c \). This results in a fine grid, denoted by \( G_f \), which will be the main grid on a new fine level \( \ell_f \) (See Figure 4.7).
Figure 4.7: Local grid refinement
In the implementation, to keep track of grid entity correspondence between a coarse level grid \( G_c \) and its tagged subgrid \( \hat{G}_c \), we maintain a list of indices for each kind of grid entity. From a coarse level grid \( G_c \), we can know whether a grid entity is tagged or not and its index in the tagged subgrid \( \hat{G}_c \). From a tagged subgrid \( \hat{G}_c \), we can determine whether the corresponding part of a fine grid entity is a node, an edge, a side or a cell, and its index in the coarse level grid \( G_c \). The correspondence between a coarse level grid \( G_c \) and its tagged subgrid \( \hat{G}_c \) on the same level \( \ell_c \) by lists of indices helps data transferring and up-/downscaling among different level grids (see Figure 4.8). This indicates we must store information about overlying coarse cell and underlying fine cells for data transfer and must recompute this information after each regrid.

A locally refined grid has two different kinds of boundaries. One kind of boundary coincides with the physical boundary of the computational domain. Another kind of boundary is in fact in the interior of the domain. For example, if the coarse grid in Figure 4.9 covers the whole computational domain, then the nodes \( A \), \( B \) and \( C \) are on the physical boundary but other three boundary nodes \( D \), \( E \) and \( F \) are in the interior of the domain. We call the second kind of boundary of a grid the \textit{coarse-fine grid interface boundary}, or simply the \textit{interface boundary}.

### 4.3.5 Hierarchy of Levels

Our AMR algorithm uses a hierarchy of properly nested levels in the sense that the main grid on a fine level is properly nested within that of its coarser level. See Figure 4.10 for a hierarchy of three properly nested levels. The highest level, denoted by level \( \ell_0 \), corresponds to the coarsest grid. The fine level resulting from refinement of a subgrid of the main rid on level \( \ell_j \) is called level \( \ell_{j+1} \). In the case that level \( \ell_j \) is not the coarsest level \( (j > 0) \), the subgrid to be refined must be a proper subgrid.
Figure 4.8: Correspondence among level grids
of the main grid on that level. By a proper subgrid, we mean that its boundary coincides with the boundary of its parent grid at most along the physical boundary of computational domain. For example, the levels in Figure 4.11 are not properly nested since the boundaries of level $\ell_1$ and level $\ell_2$ grids intersect along a vertical line segment, which is interior to the computational domain covered by the root level $\ell_0$. The three levels in Figure 4.10 are properly nested even though the boundaries of level $\ell_1$ and $\ell_2$ in Figure 4.10 (c) and (d) have a non-empty intersection, which is on the physical boundary of the computational domain.

In addition to the requirement that different levels have different grid sizes, each level has its own time step of different length for time integration. Typically, the finer the level, the smaller is the associated time step. Denote the time step on level $\ell_j$ by $\Delta t_j$ for $j \geq 0$. Then $\Delta t_{j+1} < \Delta t_j$. In this sense, the hierarchy of levels is not only spatially nested but also temporally nested.

The root level in the hierarchy is fixed since the coarsest grid on the level does not change dynamically. All other fine levels exist only temporarily due to the dynamic change of their grids. Let $\kappa$ denote the regrid interval in the algorithm and $\mathcal{L}_j$
Figure 4.10: Hierarchy of properly nested levels
Figure 4.11: Hierarchy of improperly nested levels
Figure 4.12: Hierarchy of dynamic levels \( (L_j = \kappa \Delta t_{j-1} \) represents the time duration of level \( \ell_j \) and \( \kappa \) is the regrid interval) represent the time duration of level \( \ell_j \). Then the time duration \( L_j \) of level \( \ell_j \) is given by the product of the regrid interval \( \kappa \) and the time step length \( \Delta t_{j-1} \) associated with level \( j - 1 \), i.e., \( L_j = \kappa \Delta t_{j-1} \) (See Figure 4.12).

In addition to a main grid, as indicated by the name, each level may have two additional subgrids if it has a finer level. One consists of the tagged cells, which contain large error data. It is uniformly bisected or refined to generate the main grid of the finer level. Another one consists of the untagged cells, which have small error data in contrast. It is not necessary for the latter one to exist in the AMR algorithm. But, sometimes its existence can help data transfer and up-/downscaling between coarse and fine levels.

In the implementation, levels are stored as doubly-linked lists. Each level has pointers to its coarser and finer levels. The coarser level is called the parent level;
the finer level is called the child level. A level has many other attributes other than grids. These include level index, time stamps and time steps, for example.

4.4 Elements of the Adaptive Process

4.4.1 Recursive Integration

The time integration in our AMR algorithm follows that proposed by Berger and Oliger. A coarse level is first advanced with a coarse time step. If a finer level does not exist and the refinement has not reached the maximum level specified by the user, a finer level is created. If a finer level exists and it is time for the finer level to regrid, the finer level is regridded. Next, the finer level is advanced with a fine time step. During refining or regridding of a fine level, an error estimation procedure based on the previously one-step advanced data on the coarse level is used to tag cells for refinement. The three steps: advancing, estimating and regridding, make up the recursive time integration in the AMR algorithm. Figure 4.13 shows the recursive time integration of a three level system with regrid interval $\kappa = 2$.

4.4.2 Timestep Selection

Data from both coarse and fine levels have to be synchronized before regridding a fine level. This first requires that the time step on a fine level must be selected such that the time step $\Delta t_c$ on the coarser level is an integer multiple of the fine time step $\Delta t_f$. Second, the number of fine time steps corresponding to a coarse time step must be an integer multiple of the regrid interval $\kappa$ as well.

The time step on each level is first determined by the relevant grid and the specific problem to be solved. Then it is reduced so that synchronization with the coarser grid occurs in an integer number, multiple of the regrid interval $\kappa$, of time steps.
Integrate level $\ell_j$:

1. Advance level $\ell_j$ with a coarse time step $\Delta t_j$.

2. Infrequently Estimate errors, tag cells of level $\ell_j$ grid and uniformly refine the tagged subgrid on level $\ell_j$ to generate level $\ell_{j+1}$. Downscale data from coarse to fine level after regridding or refinement.

3. Select a fine time step $\Delta t_{j+1}$ and recursively integrate level $\ell_{j+1}$ until synchronized with the coarse level $\ell_j$. Upscale data from level $\ell_{j+1}$ to level $\ell_j$ upon synchronization.

(b) Advance $\rightarrow$ Estimate $\rightarrow$ Regrid $\rightarrow$ Advance $\rightarrow \cdots$

Figure 4.13: Recursive time integration
4.4.3 Regridding of Fine Levels

In the AMR algorithm, regridding occurs on all coarse levels. It is performed once every regrid interval number of coarse time steps. The value of the regrid interval can in principle be any positive integer. In our algorithm it may be a factor of refinement ratio $r$ or not. Since fine levels are regridded infrequently to reduce the overhead cost of regridding, we require that the regrid interval is at least $\kappa \geq 2$. For the reaction-diffusion equations from computational cardiology, the regrid interval is typically chosen as $\kappa = 2$. For the three level system in Figure 4.13 (a), level $\ell_2$ will be regridded after the $7^{th}$ and $14^{th}$ time steps. Level $\ell_1$ will only be regridded after the $14^{th}$ time step. Level $\ell_0$ is the root level. It will never be regridded.

We completely regrid a fine level once every regrid interval number of coarse time steps. Before regridding a fine level, data on the fine level is first upscaled to its coarser level since the data from both of the coarse and fine levels have been synchronized and data from fine level is preferred over that from coarse level by assumption. Then, the coarse level advances its data by one coarse time step with the upscaled data as initial conditions. After that, based on the one-step advanced data on the coarse level, an error estimation procedure is employed to tag cells for refinement. In the meantime, those tagged cells on the coarse level are re-arranged to form a subgrid. Finally, we obtain a fine grid, which will be the main grid of a new fine level, by uniformly refining the subgrid on the coarse level.

Note that some fine cells are both in old and new fine grids. After the regridding of a fine level, the data associated with these cells on the old fine level must be transferred to the new fine level before the old fine level is deleted. Data transferring between two fine levels of the same depth will be discussed in section 4.5.
4.4.4 Tagging of Coarse Cells

Assume a cell-based error estimator is used to tag cells in the main grid $G_c$ on a coarse level $\ell_j$. If a cell is tagged for refining, all of its lower dimensional entities (sides, edges, etc.) on the cell must also be tagged. Here, it is implied that there is a boolean tag corresponding to each entity in the grid representation. Then we may identify and index the tagged entities by looping over each list of grid entities in the coarse grid $G_c$. This will produce a tagged subgrid, denoted by $\tilde{G}_c$.

In some cases, we may use a node-, edge- or side-based error estimator for tagging of cells. If a node, edge or side is tagged for refining, all cells which are directly connected to the node, edge or side are also tagged for refining. After that, we follow the procedures above to tag other grid entities. Ultimately, the regridding process works with cell tags.

The tagged subgrid $\tilde{G}_c$ above is not the right one to be uniformly refined to generate a fine level grid. Note that we choose to regrid fine levels infrequently. If the region covered by a fine level grid is not large enough, large error data may go beyond it before the fine level $\ell_{j+1}$ is regridded next time. For the sake of reliability and effectiveness of the AMR algorithm, a few additional layers of buffer cells need to be added to the initially tagged subgrid $\tilde{G}_c$. That is, all cells sufficiently near the subgrid $\tilde{G}_c$ are also tagged. The buffer of these cells is called the “error buffer” or “error cells”. The width of this buffer has been chosen to be equal to the regrid interval minus one, i.e., $\kappa - 1$, since error estimation is performed after the coarse level $\ell_j$ is advanced by one time step $\Delta t_j$ and there are still $\kappa - 1$ coarse time steps left before the next time to regrid.

Now suppose that a fine level $\ell_{j+1}$ is created by refining the tagged subgrid buffered by the error cells, and finer levels $\ell_i$ ($i \geq j + 1$) are recursively generated, advanced
and regirded. Note that as the fine level $\ell_{j+1}$ is advanced with its fine time step $\Delta t_{j+1}$, large error data may finally arrive at the cells on the coarse fine interface boundary before the fine level $\ell_{j+1}$ is regirded next time.

In case that a finer level $\ell_{j+2}$ exists, the finer level $\ell_{j+2}$ has to be advanced and regirded as level $\ell_{j+1}$ but with a smaller time step $\Delta t_{j+2}$. Now let us consider the last regirding of the finer level $\ell_{j+2}$ right before the fine level $\ell_{j+1}$ is regirded. In order to generate the main grid for the finer level $\ell_{j+2}$, we have to tag those cells with large error data on the fine level $\ell_{j+1}$. However, some of the cells may have been on the interface boundary between level $\ell_{j}$ and level $\ell_{j+1}$. By the proper nesting assumption, any cell on an interface boundary should not be added into a subgrid for regirding the finer level $\ell_{j+2}$. This yields a contradiction. It seems that we should add more cells into the initially tagged subgrid $\tilde{G}_c$ on the coarse level $\ell_{j}$. Actually, in the AMR algorithm, one additional layer of buffer cells, called the “nesting buffer” or “nesting cells”, are added to the subgrid above. Now the total number of layers of buffer cells is equal to the regrid interval $\kappa$. That is, regirding an intermediate fine level requires $\kappa$ layers of cells to buffer the initially tagged subgrid $\tilde{G}_c$. For an illustration, see Figure 4.14.

In Figure 4.14, we assume the AMR algorithm is applied to solve a 1D problem involving a traveling wave with speed one. The time step size in each level is selected to be equal to the mesh width. We also assume that, for simplicity, the algorithm only tags the cells where the wave front lies. Suppose we choose the refinement ratio to be $r = 2$ and the regrid interval to be $\kappa = 2$. During tagging of cells, if we only buffer $\kappa - 1 = 1$ layer of cells, only three cells are tagged for further refinement. In the Figure, the tagged and buffered cells are in thick lines and the positions where the traveling wave front is located are marked by bullets. The coarse level $\ell_{j}$ tags its cells at $t + \Delta t_{j}$ to regird the fine level $\ell_{j+1}$. The fine level $\ell_{j+1}$ tags its cells at
Figure 4.14: Improper tagging of cells (regrid interval $\kappa = 2$ and refinement ratio $r = 2$)

$t + 3\Delta t_j/2$ to regrid its finer level $\ell_{j+2}$, if it exists. In the Figure, we see that the right most cell on the finer level $\ell_{j+2}$ is right next to a cell on the coarse level $\ell_j$. This violates the proper nesting assumption for the AMR algorithm. So, if the finer level $\ell_{j+2}$ exists, buffering only $\kappa - 1 = 1$ layer of error cells is not enough for the proper nesting assumption not to be violated. Instead, we need to buffer one more layer of cells during regridding level $\ell_{j+1}$.

In case that a finer level $\ell_{j+2}$ does not exist (i.e., the fine level $\ell_{j+1}$ is the finest level in the system), there is no need to add more cells to the buffered grid of $\check{G}_c$ with buffer cells. In other words, regridding a finest level requires only $\kappa - 1$ layers of cells to buffer the initially tagged subgrid $\check{G}_c$.

It is the buffered grid of $\check{G}_c$ that will be uniformly refined to generate a fine grid $G_f$. The extended subgrid is denoted by $\hat{G}_c$ as shown in Figure 4.7. It is called the tagged subgrid $\hat{G}_c$ on a coarse level.
4.4.5 Data Up/Down Scaling

It is necessary to transfer data between coarse and fine levels in the AMR algorithm. In the case that a fine level is just created, data on the level is either initialized with initial conditions or downscaled from that on its coarser level. For the reaction-diffusion problems with piecewise linear finite elements, data downscaling is implemented by linear interpolation from node-centered data on the coarse level to that on the fine level.

Since the AMR algorithm assumes that data on a fine level grid is better than that on a coarse level grid, fine data have to be upscaled to the coarse level upon synchronization. To upscale node-centered data, we simply copy data from a fine to a coarse level if a fine grid node matches a coarse grid node. To upscale cell-centered data, for example for hyperbolic conservation laws, we replace the data on...
a coarse cell with a volume weighted average of data on the fine cells generated from refinement of the coarse cell.

In the three level system of Figure 4.13 (a), after the 7th and 14th time steps, the fine data from level \( \ell_2 \) must be upscaled to level \( \ell_1 \) upon synchronization of the coarse and fine levels. The fine data from level \( \ell_1 \) must be upscaled to level \( \ell_0 \) upon synchronization of level \( \ell_0 \) and level \( \ell_1 \). Data upscaling occurs only when coarse and fine levels are synchronized. In Figure 4.15, level \( \ell_2^{(2)} \) upscales its data to level \( \ell_1^{(1)} \) at time \( t + 2\mathcal{L}_2 = t + \mathcal{L}_1 \), which implies that \( \mathcal{L}_1 = 2\mathcal{L}_2 \).

In our algorithm, data up-/downscaling can be easily implemented by the hierarchical ordering of grid entities since the grid entity correspondence between coarse and fine levels is well-defined through the tagged subgrid on coarse level (see Figure 4.8).

### 4.4.6 Interface Data Correction

Sometimes it is necessary to make coarse-fine interface data correction after data on a fine level is upscaled to its coarser level upon synchronization.

For the conservation laws, the cell-centered data on coarse cells adjacent to fine grid must be corrected to maintain conservation and free-stream preservation for the sake of algorithm accuracy and stability.

For the reaction-diffusion problems, if the node-centered data on the fine level side along a coarse-fine interface are integrated with fine time steps before upscaling, then the data on the coarse-fine composite grid are not continuous due to the nonlinearity of reactions. In the meantime, interface data correction must be performed such that the data are continuous through hanging nodes.
4.5 Initial and Boundary Conditions

In the case that the AMR algorithm is applied to time-dependent problems, both initial and boundary conditions must be implemented correctly.

Recall that a locally refined grid has two different kinds of boundaries: the physical boundary and the interface boundary. Data on the physical boundary are set by applying boundary conditions for the problem to be solved. Data on the interface boundary usually involves communication between coarse and fine grids. Interface boundary data on the fine grid side may be copied or interpolated from the coarse grid side. Interface boundary data on the coarse grid side may be copied or restricted from the fine grid side. The hierarchical ordering with boundary entities ordered first plays an important role in those situations.

If a level is at the initial time, data on the level can be simply initialized with supplied initial conditions for the problem to be solved. However, most levels, which are dynamically created in the middle of time integration, are not at the initial time. Data associated with them should be initialized in a different way.

Note that the main grid on a fine level is generated by uniform refinement of a tagged subgrid of that on a coarse level. It is natural that data on a fine level is initialized with data downscaled from the coarse level.

However, in case that an old fine level exists and the current fine level is generated by regridding, only downscaling data from coarse to fine level is not enough. Some data on the old fine level must be transfered to the current fine level. In Figure 4.15, after level $\ell_1^{(2)}$ is created, some data must be copied from level $\ell_1^{(1)}$, which shares the same parent level with level $\ell_1^{(2)}$. After level $\ell_2^{(3)}$ is created, some data must be copied from level $\ell_2^{(2)}$, which has a different parent level from level $\ell_2^{(3)}$. In the AMR algorithm, the pair of a current fine level and its old fine level of the same depth are
called the “sibling levels”.

Two sibling levels may share a single parent coarse level or have different parent coarse levels. In the three level system shown in Figure 4.15, level \( \ell_1^{(1)} \) and level \( \ell_1^{(2)} \) shares the same parent coarse level \( \ell_0 \). Level \( \ell_2^{(2)} \) and level \( \ell_2^{(3)} \) have different parent coarse levels.

In case that two sibling levels share a single coarse level, both levels have an one-to-one correspondence with the parent coarse level. A correspondence can then be naturally built up between these two levels. For example, suppose we are given a node on the current fine level, which corresponds to an edge in the parent coarse level. From the parent level, we can check if the edge is refined in the previous regridding or not. If it is, the index of the node corresponding to the edge on the old fine level can be obtained easily. Then data associated with the node on the old fine level can be copied by the index to the current fine level. In the implementation, similar to the correspondence between a level grid and its tagged subgrid, a few lists of indices to sibling level are maintained for each fine level. If an entity on the current level can not find its corresponding one on the old fine level, the associated index in the lists is set as empty or initialized with a negative number.

In case that two sibling levels have different parent coarse levels, the two parent coarse levels must be a pair of sibling levels by the construction of levels. There are a few lists of indices to its sibling level on the newer one of the two parent levels. Naturally, the correspondence between the two fine sibling levels can be built up through the correspondence between the two parent coarse levels. For example, level \( \ell_2^{(3)} \) in Figure 4.15 first looks up in level \( \ell_1^{(2)} \), then checks with level \( \ell_1^{(1)} \) and finally finds out its sibling indices on level \( \ell_2^{(2)} \).

In a word, data on a fine level is first initialized with data downscaled from its parent coarse level. Then if an old fine level exists, some data are copied from the
sibling level.

4.6 Error Estimation

As discussed, an error estimation procedure must be provided for tagging of cells during local grid refinement. In principle, the AMR algorithm supports other error estimation strategies such as the explicit residual estimators, the implicit estimators based on local Neumann or Dirichlet problems, the hierarchical estimators, the recovery based error estimators and the interpolation based error estimators [8, 137, 186, 187]. We consider that a posteriori error estimates usually require problem-dependent analysis and coding [109, 122], which may be very difficult especially for the monodomain/bidomain model problems with complex membrane dynamics. So, we would rather employ an error estimation strategy which is independent of the partial differential equations to be integrated and the numerical methods used.

Following the strategy proposed by Berger-Oilger and Trangenstein for the block-structured AMR algorithm [181], we use both Richardson extrapolation to estimate the local truncation error in the integration, and a simple device to estimate the number of time steps to be performed on the current level. This method is standard in the numerical integration of ordinary differential equations [63].

Suppose that at each time step we commit an error of magnitude \( \epsilon \) (principally the local truncation error); further, suppose that the computation permits a bound \( M \) on the growth of these errors. Note that a computational solution essentially amounts to applying a perturbation of the identity operator to the previous solution. It is therefore reasonable to expect \( M \) to be close to 1 for smooth solution and sufficiently
fine grid. Then the error $e_n$ at step $n$ satisfies
\[ e_1 \leq \epsilon, \quad e_n \leq \epsilon + M e_{n-1} \quad \forall \ n > 1. \]

Then an argument by induction shows that
\[ e_n \leq \epsilon \sum_{j=0}^{n-1} M^j = \epsilon \frac{1 - M^n}{1 - M}. \]

If $M$ is close to one, then for small $n$ the error bound will be approximately $n \epsilon$, i.e.,
\[ e_n \approx n \epsilon, \quad (4.1) \]
which is linearly proportional to the number of time steps. Replacing the integer $n$ with a real number $s = \frac{t}{\Delta t}$ in the approximation (4.1) above, we may think the local error $e_s$ as a linear function of time $t$, i.e.,
\[ e(t) \equiv e_s = s \epsilon = \frac{t}{\Delta t} \epsilon, \]
for $t$ on the order of the time step $\Delta t$ (not too large and not too small).

Suppose that the local truncation error satisfies
\[ \epsilon = C \cdot \Delta t^{p+1}, \]
where $p$ is the expected global order of the scheme. Here, we are assuming that spatial and temporal error orders are equal. Then after integrating by one coarse step of size $r \Delta t$, the error at time $\nu \Delta t$ is given by
\[ e^c(\nu \Delta t) \approx \frac{\nu \Delta t}{r \Delta t} \cdot C \cdot (r \Delta t)^{p+1} = \nu \cdot C \cdot r^p \Delta t^{p+1}. \]

Here, $\nu$ is the number of fine time steps and it must be an integer multiple of the regrid interval $\kappa$ plus one in the case that the regrid interval $\kappa$ is unequal to the
refinement ratio \( r \) (say, \( \kappa = 2 \) and \( r = 4 \)). On the other hand, if we take \( \nu \) fine time steps of size \( \Delta t \), the accumulated error \( e^f(\nu \Delta t) \) is approximately equal to the number \( \nu \) of fine time steps times the local truncation error \( \epsilon \), i.e.,

\[
e^f(\nu \Delta t) \approx \nu \cdot C \cdot \Delta t^{p+1}.
\]

This allows us to approximately compute the local truncation error of a fine time step by

\[
\epsilon \approx \frac{e^c(\nu \Delta t) - e^f(\nu \Delta t)}{(r^p - 1) \cdot \nu} = \frac{w^c(\nu \Delta t) - w^f(\nu \Delta t)}{(r^p - 1) \cdot \nu},
\]

where \( w \) is the quantity being monitored for errors. Then the global error can be estimated by multiplying the local error (4.2) by an anticipated number \( N \) of time steps. We approximate

\[
N \approx \frac{\mathcal{L}}{\lambda \Delta t},
\]

where \( \mathcal{L} \) is some length scale associated with the problem to be solved, \( \lambda \) is some important wavespeed, and \( \Delta t \) is the current time step. Moreover, a relative (global) error can be estimated by dividing the global error by an appropriate quantity such as the maximum of the monitored data. In the algorithm, a cell is tagged if the relative error is greater than a specified tolerance, i.e.,

\[
\frac{|w^c(\nu \Delta t) - w^f(\nu \Delta t)|}{\max |w^f(\nu \Delta t)|} \cdot \frac{\mathcal{L}}{(r^p - 1) \cdot \nu \cdot \lambda \cdot \Delta t} > \text{tolerance}.
\]

It is noteworthy that the data used in (4.3) are all available and the errors can be estimated without too much additional cost. In this sense, the Richardson extrapolation (actually interpolation) based error estimation procedure is cheap.

**Remark 4.1:** The time interpolation used to obtain the monitored quantities \( w^c(\nu \Delta t) \) in (4.3) should have accuracy at least one order higher than the overall
algorithm in accuracy. Otherwise, the interpolation error may affect (enlarge) the size of the refined region. The strategies used by Berger-Oliger and Trangenstein avoid this problem [24, 27].

In Figure 4.16, to regrid level $\ell_{j+1}$ at time $t$, data from both level $\ell_j$ and level $\ell_{j-1}$ at time $t + \Delta t_j$ are compared to compute relative errors by that in (4.3). To regrid level $\ell_{j+2}$ at time $t$, data from level $\ell_{j+1}$ and level $\ell_j$ at time $t + \Delta t_{j+1}$ are used to estimate errors. To regrid level $\ell_{j+2}$ at time $t + 2\Delta t_{j+1}$, data from level $\ell_{j+1}$ and level $\ell_j$ at time $t + 3\Delta t_{j+1}$ are used in the error estimation procedure.

The Richardson extrapolation based error estimation procedure is a two-level method. This implies that it needs a coarse level and a coarser level for regridding a fine level. In other words, in order to regrid level $\ell_{j+1}$, both level $\ell_{j-1}$ and level $\ell_j$ must exist and be active (This is different from Trangenstein’s approach). This may

Figure 4.16: Error estimation (regrid interval $\kappa = 2$ and refinement ratio $r = 2$)
give us an impression that the Richardson extrapolation error estimation procedure will be very expensive. As a matter of fact, the data on the coarse levels $\ell_{j-1}$ and $\ell_j$ are always easily available for the error estimation since each of the coarse levels must have been advanced once by its own coarse time step before the fine level $\ell_{j+1}$ is scheduled to be regridded. To compute the relative error by (4.3), what the two-level procedure need to do is as simple as copying and interpolating of coarse data.

Of course, we have to admit that, when $j = 0$ in Figure 4.16, level $\ell_{-1}$ does not exist for regridding level $\ell_1$. In this case, the Richardson extrapolation based error estimator can not be applied. We should instead use a single-level error estimator such as a gradient detector, or uniformly refine the root level $\ell_0$ to generate level $\ell_1$ at the beginning of time integration.

In the meantime, even if the second level $\ell_1$ is generated from uniform refinement of the root level grid $G_0$, the cost of uniformly refining and integrating the root level $\ell_0$ is still relatively small since, by the assumption for our AMR algorithm, the coarsest grid $G_0$ usually has only a small number of cells and the uniformly refined grid $G_1$ on the second level $\ell_1$ hence would also not have too many elements, which implies that just little extra work is involved.

In addition, for the error estimation, we only need to know the order $p$ of the local truncation error $\epsilon$ and not its exact form (in terms of derivatives of the solution etc.). The estimate inequality (4.3) is dimensionless and independent of the numerical methods used and the partial differential equations being integrated. In this sense, the Richardson extrapolation based error estimation procedure above is very general as well as simple and cheap.
Chapter 5

Multilevel/Multigrid Iterations

The multilevel/multigrid iteration methods are standard in modern scientific computing. In section 5.1, a brief introduction to the multigrid methods is presented. Since the standard multigrid iteration methods can be formulated under the framework of subspace corrections, we next discuss some abstract subspace correction algorithms in section 5.2. Then the standard V-cycle multigrid algorithm is described in section 5.3 and a short convergence analysis for the algorithm is also presented there. We discuss in greater detail the two building blocks, the fine grid smoothing and the subspace (coarse grid) correction, of the standard V-cycle multigrid algorithm on uniformly and locally refined grids in section 5.4 and section 5.5, respectively. In section 5.6, a few more implementation issues, including the initial guess, the coarsest grid solver and the iteration stopping criterion, are also discussed. In the last part of this chapter, some relative and absolute error estimates are proposed for terminating the multigrid algorithm.

5.1 Introduction

As discussed in Chapter 3, finite element discretization of the linear diffusion resulting from an operator splitting process for a singularly perturbed reaction-diffusion system yields a linear system in each time step. In the case of the monodomain model, the size of the linear system is equal to the number of nodes in a partition of the domain Ω; in the case of the bidomain model, it is twice the number of nodes. Usually the linear system is very large, involving hundreds of thousands or millions of unknowns. To efficiently solve such linear systems, a fast iterative method must be applied.
It is well-known that the standard multilevel/multigrid methods are among the most efficient and optimal iterative algorithms. For $H^2$-regular problems, a multigrid iteration only involves linear complexity and it reduces the energy norm of the approximation error at least by a factor which is uniformly less than one regardless of the number of levels and mesh parameters. The overall work is at most linearly proportional to the total number of unknowns in the linear system to be solved.

A standard multigrid method has two main features: smoothing and subspace correction. The smoothing procedure has the effect of efficiently damping out the high frequency (oscillatory) part of the iteration error while the subspace correction quickly reduces the low frequency (smooth) component. Any convergent iteration, such as Richardson, weighted Jacobi, Gauss-Seidel or Successive Over-Relaxation (SOR), can be used as a smoother in a multigrid algorithm. Relatively, the subspace correction part is a little bit more complicated. A subspace correction may be based on the structure of algebraic equations or the topology of underlying coarse grids. The corresponding multigrid iteration is called an algebraic or a geometric multigrid. Since our AMR algorithm uses a hierarchy of properly nested levels, the underlying coarse grid for a fine grid is always available for subspace correction. It is easier to apply the geometric multigrid algorithms than the algebraic ones. So, in the thesis work, we are mainly interested in the geometric multigrid iterations.

A multigrid iteration on locally refined grids is also called a composite grid iteration. It proceeds recursively on a grid hierarchy and works with at most two grids, correspondingly a fine subdomain and a coarse subdomain, in any region of space at any time. There are a variety of composite grid iteration algorithms, such as the FAC method [124], the BEPS multigrid method [38], the hierarchical basis multigrid method [14], the BPX multilevel nodal basis method [44] and the cascadic multigrid method [30]. It is straightforward that the FAC and the BEPS composite
grid iteration methods can be incorporated into the implementation of our AMR algorithm.

The FAC composite grid iteration first makes a subspace correction on the fine subdomain; after the solution is updated, compute the residual; then make another subspace correction on the coarse domain.

The BEPS composite grid iteration splits a general elliptic problem on the fine subdomain into two sub-problems. One sub-problem has a non-homogeneous right hand side with homogeneous boundary conditions; another sub-problem has a homogeneous right hand side with non-homogeneous boundary conditions. First, the iteration approximately solves the first sub-problem on the fine subdomain. Second, after restricting the residual from the fine subdomain to the coarse subdomain, it approximately solves a coarse grid problem on the coarse subdomain. Third it solves the second sub-problem on the fine subdomain, where the solution is used as a correction to that obtained in the first step.

In the thesis work, for simplicity, we only consider the application of the standard V-cycle multilevel/multigrid algorithm to solving linear systems. The algorithm and its convergence analysis are briefly presented in this chapter under the framework of subspace corrections. For the general and detailed theory of multilevel/multigrid methods, we recommend the books [37, 87, 125, 140, 191, 200] and the references therein.

5.2 Subspace Correction Methods

Considering that the standard multigrid/multilevel iteration is a special subspace correction technique, we would like to discuss the general method first. For details of subspace correction methods, refer to [36, 43, 83, 195, 196, 200]. The discussion in this and the next sections mainly follows [200].
A subspace correction method is actually a preconditioning strategy for iteratively solving linear systems. It is well known that it is not efficient to make error correction for a linear system on the same space all the times. For example, the weighted Jacobi, Gauss-Seidel or Successive Over-Relaxation iteration for solving a large linear system may converge to the solution extremely slowly, even if the linear system is symmetric and positive definite. In contrast, it is demonstrated by both theoretical analysis and numerical experiments that making error correction only on a subspace of the original large space can yield very good results in the sense that low frequency errors, which can not be efficiently eliminated in the original space, can be damped out very quickly.

Now let us consider the following general discrete variational problem in a finite element space $V$:

$$\text{Find } u \in V, \text{ such that}$$

$$a(u, v) = f(v) \quad \forall \ v \in V. \quad (5.1)$$

Here, $a(u, v) : V \times V \to \mathbb{R}$ is a symmetric, bounded and coercive bilinear form; $f(v) : V \to \mathbb{R}$ is a continuous linear functional. By the Lax-Milgram theorem, the variational problem (5.1) has a unique solution [138].

After introducing a symmetric and positive definite operator $A : V \to V$ by the relation (definition)

$$(A u, v) \equiv a(u, v) \quad \forall \ u, v \in V, \quad (5.2)$$

the problem (5.1) can be equivalently written as an abstract linear equation

$$Au = f. \quad (5.3)$$

Let $W$ be a subspace of the finite element space $V$. Then the variational problem (5.1) restricted to $W$ reads
Find $w \in W$, such that

$$a(w, v) = f(v) \quad \forall v \in W. \quad (5.4)$$

Similarly, we can introduce a symmetric and positive definite operator $A_W : W \to W$ by the relation (definition)

$$(A_W w, v) \equiv a(w, v) \quad \forall w, v \in W. \quad (5.5)$$

The problem (5.4) is equivalent to another abstract linear equation

$$A_W w = f. \quad (5.6)$$

Here, the operator $A_W$ is called the Ritz approximation of the operator $A$ with respect to the subspace $W$.

In addition, we need two kinds of orthogonal projections from $V$ onto the subspace $W$. The projection $Q_W : V \to W$ is defined by

$$(Q_W u, v) = (u, v) \quad \forall v \in W, \quad (5.7)$$

and the projection $P_W : V \to W$ by

$$a(P_W u, v) = a(u, v) \quad \forall v \in W. \quad (5.8)$$

The first one $Q_W$ is called the $L^2$-orthogonal projection and the second one $P_W$ is called the $a$-orthogonal projection.

To solve for $u \in V$ in (5.3), we apply an iterative procedure by the following subspace correction:

$$\tilde{u} \leftarrow \tilde{u} + P_W (u - \tilde{u}). \quad (5.9)$$

The subspace correction (5.9) makes the approximation error

$$\tilde{e} = u - \tilde{u}$$

128
Orthogonal to the subspace $W$.

Note that the operators $A$, $A_W$, $P_W$ and $Q_W$ are connected by the relation
\[ A_W P_W = Q_W A \]  
(5.10)
since
\[ (A_W P_W u, v) = a(P_W u, v) = a(u, v) = (A u, v) = (Q_W A u, v). \]

Then the subspace correction (5.9) can be rewritten as
\[ \tilde{u} \leftarrow \tilde{u} + A_W^{-1} Q_W A (u - \tilde{u}), \]
or
\[ \tilde{u} \leftarrow \tilde{u} + A_W^{-1} Q_W (f - A \tilde{u}). \]  
(5.11)
The subspace correction (5.11) is far too expensive to be a reasonable method if the subspace $W$ is not sufficiently small or too complicated. In practice, it should be replaced by an approximate subspace correction:
\[ \tilde{u} \leftarrow \tilde{u} + B_W^{-1} Q_W (f - A \tilde{u}), \]  
(5.12)
with a symmetric and positive definite operator $B_W : W \to W$.

The operator $B_W$ should have the property that the correction term
\[ d_W = B_W^{-1} Q_W (f - A \tilde{u}) \]
can be easily computed as the solution of the linear system
\[ (B_W d_W, v) = (f - A \tilde{u}, v) \equiv (f, v) - a(\tilde{u}, v) \quad \forall v \in W. \]
The computation may consists of several steps of a given convergent iterative procedure:
\[ \tilde{w} \leftarrow \tilde{w} + R_W (r_W - A_W \tilde{w}) \]
for the linear system
\[ A_W w = r_W \]
where \( r_W = Q_W (f - A \tilde{u}) \). The relaxation operator \( R_W \) can be a weighted Jacobi iteration, a symmetric Gauss-Seidel, a symmetric Successive Over-Relaxation or any other symmetric convergent iteration. Let \( \nu \) be the step number in the procedure. Then the operator \( B_W \) in the approximate subspace correction (5.12) is defined by

\[ B_W^{-1} = (I - (I - R_W A_W)^\nu)A_W^{-1} \]  

(5.13)
since the error propagation (reduction) operator

\[ I - B_W^{-1}A_W^{-1} = (I - R_W A_W)^\nu. \]

Here, both operators \( B_W \) and \( B_W^{-1}Q_W \) are symmetric and positive definite since

\[ B_W^{-1}Q_W = Q_W B_W^{-1}Q_W. \]

As a matter of fact, different subspaces can be simultaneously or sequentially used to make (approximate) subspace corrections. Let \( W_0, W_1, \ldots, W_J \) be subspaces of \( V \). We usually assume that every \( v \in V \) can be written as

\[ v = w_0 + w_1 + \cdots + w_J, \]  

(5.14)
with \( w_j \in W_j \) \((j = 0, 1, \ldots, J)\).

If the subspace corrections are made simultaneously, i.e.,

\[ \tilde{u} \leftarrow \tilde{u} + \sum_{j=0}^J B_W^{-1}Q_W (f - A \tilde{u}), \]  

(5.15)
one obtains an additive subspace correction method corresponding to the subspaces \( W_0, W_1, \ldots, W_J \) of \( V \). In an additive subspace correction, the approximation error \( \tilde{e} = u - \tilde{u} \) is corrected by

\[ \tilde{e} \leftarrow \left( I - \sum_{j=0}^J T_j \right) \tilde{e}, \]
where $T_j = B_{W_j}^{-1}Q_{W_j}A$. It is equivalent to preconditioning the linear equation (5.3) by

$$\tilde{u} \leftarrow \tilde{u} + B^{-1}(f - A\tilde{u})$$

with a symmetric and positive definite preconditioner

$$B^{-1} = \sum_{j=0}^{J} B_{W_j}^{-1}Q_{W_j}.$$  

If the subspace corrections

$$\tilde{u} \leftarrow \tilde{u} + B_{W_j}^{-1}Q_{W_j}(f - A\tilde{u}) \quad j = 0, 1, \ldots, J \quad (5.16)$$

are made sequentially on the order $j = J, \ldots, 1, 0$, one obtains a multiplicative subspace correction method corresponding to the subspaces $W_0, W_1, \ldots, W_J$ of $V_h$. In a multiplicative subspace correction, the approximation error $\tilde{e} = u - \tilde{u}$ is corrected by

$$\tilde{e} \leftarrow \left(\prod_{j=0}^{J}(I - T_j)\right)\tilde{e}.$$  

It is equivalent to solving the linear equation (5.3) by

$$\tilde{u} \leftarrow \tilde{u} + B^{-1}(f - A\tilde{u})$$

with a relaxation operator

$$B^{-1} = \left(I - \prod_{j=0}^{J}(I - T_j)\right)A^{-1}.$$  

Here, $B^{-1}$ is nonsymmetric. The multiplicative subspace correction has to be symmetrized before it is used as a preconditioner for others such as the conjugate gradient iterations. As an example of symmetrization, the subspaces corrections (5.16)
are made sequentially first on the order $j = J, \cdots, 0$, next made in the reverse order $j = 0, 1, \cdots, J$. The operator $B^{-1}$ in the symmetrized multiplicative subspace correction then is given by

$$B^{-1} = \left\{ I - \left( \prod_{j=J}^{0} (I - T_j) \right) \left( \prod_{j=0}^{J} (I - T_j) \right) \right\} A^{-1},$$

which is symmetric and positive definite with respect to the inner product $a(\cdot, \cdot)$.

It is well-known that an additive subspace correction method is not always convergent even for symmetric and positive definite problems while a multiplicative subspace correction method converges with a reasonable choice of subspaces $\{W_j\}_{j=0}^J$ and preconditioners $\{B_{W_j}\}_{j=0}^J$ (In order to have convergence of the iteration independent of mesh parameters, the subspaces have to satisfy certain stability and approximation properties). Nevertheless, either of the (symmetrized) subspace correction methods can be used as a preconditioner for other convergent iterations, such as the conjugate gradient methods.

As stated, in a subspace correction method, it is assumed that the subspaces $\{W_j\}_{j=0}^J$ make up a decomposition of the finite element space $V$. In fact, there are a variety of ways to partition the space $V$. We can make either a “horizontal” (non-nested) or a “vertical” (nested) decomposition. The classic overlapping/non-overlapping domain decomposition methods fall into the category of non-nested decompositions [167]. The standard multigrid iterations belong to nested decompositions. It is remarkable that a non-nested decomposition may yield a nested decomposition and a nested decomposition may give rise to a non-nested decomposition. In the next section, we will present the convergence theory of a standard multigrid algorithm using a non-nested decomposition from nested subspaces.

Before ending this section, we briefly describe three strategies to derive non-nested decompositions from nested subspaces. Assume $\{V_j\}_{j=0}^J$ is a nested sequence
of subspaces of the finite element space \( V \):

\[
V_0 \subset V_1 \subset V_2 \subset \cdots \subset V_J = V.
\]

The first strategy uses the \( a \)-orthogonal projections \( P_j : V \to V_j \) \((j = 0, 1, \ldots, J)\) to decompose the space \( V \) into

\[
W_0 = V_0 \quad \text{and} \quad W_j = \{ P_ju - P_{j-1}u \mid u \in V \}, \quad j = 1, 2, \ldots, J.
\]

With this decomposition, we can prove that a standard multigrid iteration (e.g., \( V \)-cycle or \( W \)-cycle), for \( H^2 \)-regular problems, reduces the energy norm of the approximation error at least by a factor which is uniformly less than one regardless of the number of subspaces [200].

The second one uses the nodal value interpolants \( I_j : C(\bar{\Omega}) \to V_j \) \((j = 0, 1, \ldots, J)\), defined by \((I_ju)(x) = u(x)\) for any interior node \( x \) in the triangulation associated with the subspace \( V_j \), to decompose the space \( V \) into

\[
W_0 = V_0 \quad \text{and} \quad W_j = \{ I_ju - I_{j-1}u \mid u \in V \}, \quad j = 1, 2, \ldots, J.
\]

It corresponds to the hierarchical basis methods [14, 198, 199].

In the last strategy, the \( L^2 \)-orthogonal projections \( Q_j : V \to V_j \) \((j = 0, 1, \ldots, J)\) are used to decompose the space \( V \) into

\[
W_0 = V_0 \quad \text{and} \quad W_j = \{ Q_ju - Q_{j-1}u \mid u \in V \}, \quad j = 1, 2, \ldots, J.
\]

This one corresponds to the multilevel nodal basis methods or the so-called BPX preconditioners [44].

5.3 The Standard \( V \)-cycle Multigrid Algorithm

A standard multigrid method for the solution of finite element equations like (5.3) falls into the category of multiplicative subspace correction methods. It is a multilevel
algorithm, which uses a nested sequence of subspaces of the space $V$:

$$V_0 \subset V_1 \subset V_2 \subset \cdots \subset V_J = V, \quad (5.20)$$

corresponding to a hierarchy of nested level grids. More details will be discussed later on how to construct such a nested sequence of finite element subspaces.

Let $P_j : V_{j+1} \to V_j$ and $Q_j : V_{j+1} \to V_j$ ($j = 0, 1, \cdots, J$) be the $a$-orthogonal and $L^2$-orthogonal projections, respectively; let $A_j$ ($j = 0, 1, \cdots, J$) be the Ritz approximation of the symmetric and positive definite operator $A$ in (5.2) with respect to the subspace $V_j$. That is, the operators are defined by

$$a(P_j u, v) = a(u, v) \quad \text{for } u \in V_{j+1}, \ v \in V_j. \quad (5.21)$$

$$(Q_j u, v) = (u, v) \quad \text{for } u \in V_{j+1}, \ v \in V_j, \quad (5.22)$$

and

$$(A_j u, v) \equiv a(u, v) \quad \text{for } u, v \in V_j. \quad (5.23)$$

To define the smoothing process, we require a linear operator $S_j : V_j \to V_j$ for $j = 0, 1, \cdots, J$. Assume $S_j$ is symmetric and positive definite. We define the standard $V$-cycle multigrid iteration $B_j^{-1} : V_j \to V_j$ by induction.

**Algorithm (the $V$-cycle multigrid iteration):** Let $B_0 = A_0$. For $j > 0$, assume that $B_{j-1} : V_{j-1} \to V_{j-1}$ has been defined and define $B_j^{-1} g_j$, for $g_j \in V_j$ as follows:

**Step 1.** Pre-smoothing (relaxation): set $u_j^0 = 0$ and define $u_j^l$ for $l = 1, \cdots, \nu$ by

$$u_j^l = u_j^{l-1} + S_j(g_j - A_j u_j^{l-1}). \quad (5.24)$$

**Step 2.** Subspace correction:

$$u_j^{\nu+1} = u_j^\nu + B_{j-1}^{-1} Q_{j-1}(g_j - A_j u_j^\nu). \quad (5.25)$$
Step 3. Post-smoothing (relaxation): set $B_j^{-1}g_j = u_j^{2\nu+1}$ where $u_j^l$ is defined for $l = \nu + 2, \cdots, 2\nu + 1$ by

$$u_j^l = u_j^{l-1} + S_j(g_j - A_j u_j^{l-1}). \quad (5.26)$$

Remark 5.1: In the algorithm, the symmetric smoothing operator $S_j$ may be replaced by a nonsymmetric operator. In this case, one of the two smoothing steps must use its adjoint operator $S_j^*$ instead if the algorithm is used as preconditioner for others such as the conjugate gradient iterations.

Remark 5.2: In the second step of the algorithm, if we perform subspace correction twice stead of just once, the resulting iteration is called the standard W-cycle multigrid method.

We may think of each of the pre-smoothing and the post-smoothing steps in the $V$-cycle iteration as a single relaxation

$$\tilde{u}_j \leftarrow \tilde{u}_j + R_j(g_j - A_j \tilde{u}_j), \quad (5.27)$$

where the relaxation operator $R_j$ is given by

$$R_j = (I - (I - S_j A_j)^\nu)A_j^{-1} \quad (5.28)$$

and is symmetric and positive definite.

It is easy to find that the error propagation operator $E_j = I - B_j^{-1}A_j$ in the standard $V$-cycle algorithm is the product of those from the three steps: pre-smoothing, subspace correction and post-smoothing. That is,

$$I - B_j^{-1}A_j = (I - R_j A_j)(I - B_{j-1}^{-1}Q_{j-1}A_j)(I - R_j A_j)$$

$$= (I - R_j A_j)(I - B_{j-1}^{-1}A_{j-1}P_{j-1})(I - R_j A_j).$$
Multiplying both sides of the identity above with the $a$-orthogonal operator $P_j$ defined in (5.21) and using the fact $P_{j-1}(I - P_j) = 0$, we get a recursion

$$I - B_j^{-1}A_jP_j = (I - R_jA_jP_j)(I - B_{j-1}^{-1}A_{j-1}P_{j-1})(I - R_jA_jP_j),$$

(5.30)

for $j = J, \cdots, 1, 0$. Note that $P_J = I$. A successive application of the identity above yields

$$I - B_j^{-1}A_j = (I - T_j)(I - T_{j-1}) \cdots (I - T_1)(I - T_0)$$

$$= \left( \prod_{j=i}^{0} (I - T_j) \right) \left( \prod_{j=0}^{j} (I - T_j) \right),$$

where

$$T_0 = P_0 \quad \text{and} \quad T_j = R_jA_jP_j \quad \text{for} \quad j = 1, 2, \cdots, J.$$

In the product above, $(I - T_0)^2 = I - T_0$. Obviously, the error propagation operator $I - B_j^{-1}A_j$ in the standard multigrid algorithm has exactly the same form as the multiplicative subspace correction methods, which are introduced in the previous section. It is a multiplication of all of those from relaxations in each subspace. In this sense, the standard multigrid/multilevel iteration is multiplicative.

By the formula for the error propagation operator $E_J = I - B_J^{-1}A_J$, we know the standard multigrid iteration converges if each of the relaxations $R_j$ or the smoothers $S_j$ in the algorithm converges, i.e., the spectral radius $\rho(I - T_j) < 1$. As a matter of fact, the standard multigrid method converges with any convergent smoother $S_j$ such as the Richardson, weighted Jacobi and Gauss-Seidel smoothers. The remaining problem is how fast the multigrid iteration converges.

To estimate the convergence rate of the algorithm in a classic approach, assume that the continuous variational problem corresponding to the discrete one (5.1) satisfies the regularity assumptions: the global shape-regularity of meshes and the
full $H^2$-regularity of solutions. The global shape regularity requires that the meshes must be at least quasi-uniformly refined. By the full $H^2$-regularity, we mean that the solution to the continuous variational problem has a square-integrable second-order derivatives as long as its right hand side is square-integrable. For parabolic equations, the solution is $C^\infty$ for $t > 0$, but the Sobolev norm $\|u\|_{H^2}$ could be large, especially with reactions acting as source terms. The regularity assumptions imply that the following error estimate can be obtained [35, 196]

$$\|(I - P_{j-1}) v\|_a^2 \leq \frac{C_1}{\rho(A_j)} \|A_j v\|_a^2 \quad \forall \ v \in V_j$$

(5.32)

where $C_1$ is a constant independent of $j$, $\rho(A_j)$ is the spectral radius of $A_j$, $\|\cdot\|_a$ is the inner product $a(\cdot, \cdot)$ induced norm and $\|\cdot\|$ is the $L^2$ norm.

Let us consider the case that the smoothers $S_j$ are given by the weighted Jacobi iteration or the symmetric Gauss-Seidel point iteration. In this case, under the regularity assumptions, it can be proved that the smoothers $S_j$ satisfy the following property [39]

$$\frac{\mu}{\rho(A_j)} \|v\|^2 \leq (S_j v, v) \leq \frac{\tilde{\mu}}{\rho(A_j)} \|v\|^2 \quad \forall \ v \in V_j$$

where $0 < \mu < \tilde{\mu} < 2$ are two constants independent of $j$. Furthermore, it is not difficult to prove that the symmetric and positive definite relaxations $R_j$ defined by (5.28) ($\nu > 1$) satisfy the approximation property [43]

$$\frac{C_2}{\rho(A_j)} \|v\|^2 \leq (R_j v, v) \quad \forall \ v \in V_j$$

(5.33)

where $C_2$ is a constant independent of $j$.

In addition, we assume that the relaxation error operators $I - R_j A_j$ are symmetric and positive definite with respect to the inner product $a(\cdot, \cdot)$. This assumption is
equivalent to say that the relaxation operators $R_j$ satisfy the smoothing property

$$\langle R_j v, v \rangle \leq \tilde{C}_2 \langle A_j^{-1} v, v \rangle \quad \forall v \in V_j,$$

(5.34)

for some constant $\tilde{C}_2 \in (0, 1)$. This property holds if the smoothers $S_j$ are given by the weighted Jacobi iteration or the symmetric Gauss-Seidel iteration.

Denote by $K_j$ the relaxation error operator $I - R_j A_j$ for $j > 0$. Assume $K_0 = I$. So, all $K_j$ ($j \geq 0$) are invertible. Let $\hat{E}_j = K_j^{-1} E_j$ for $j \geq 0$. Recall that the error propagation operators $E_j = I - B_j^{-1} A_j$ of the V-cycle multigrid algorithm have the following recursion relation

$$E_j = K_j (I - B_{j-1}^{-1} A_{j-1} P_{j-1}) K_j.$$

We can write the error propagation operator $E_j$ in a different way

$$E_j = K_j (I - (I - E_{j-1}) P_{j-1}) K_j$$

$$= K_j ((I - P_{j-1}) + E_{j-1} P_{j-1}) K_j$$

$$= K_j (I - P_{j-1}) K_j + K_j E_{j-1} P_{j-1} K_j,$$

or equivalently,

$$\hat{E}_j = (I - P_{j-1}) K_j + K_{j-1} \hat{E}_{j-1} P_{j-1} K_j.$$

(5.36)

For any $v \in V_j$, by the orthogonality of $P_{j-1}$, the following identity holds

$$\| \hat{E}_j v \|^2_a = \| (I - P_{j-1}) K_j v \|^2_a + \| \hat{E}_{j-1} P_{j-1} K_j v \|^2_a.$$

(5.37)

Here, $\| (I - P_{j-1}) K_j v \|^2_a$ corresponds to the high frequency part of the error and $\| \hat{E}_{j-1} P_{j-1} K_j v \|^2_a$ corresponds to the low frequency part.
It follows from (5.32) and (5.33) that, for \( j > 0, \)
\[
\| (I - P_{j-1})K_jv \|^2_a \leq \frac{C_1}{\rho(A_j)} \| A_jK_jv \|^2_a
\]
\[
\leq \frac{C_1}{C_2}(R_jA_jK_jv, A_jK_jv)
\]
\[
= \frac{C_1}{C_2}((I - K_j)K_j^2v, v)_a
\]
\[
\leq \frac{C_1}{2C_2}(\|v\|^2_a - \|K_jv\|^2_a)
\]
since \( K_j \) is symmetric and positive definite with respect to the inner product \( a(\cdot, \cdot) \).

Let \( \delta = C_1/(C_1 + 2C_2) \). We will prove the following estimate by induction
\[
\| \hat{E}_j \|^2_a \leq \delta \quad \text{for } j = 0, 1, \cdots, J. \tag{5.39}
\]
It is obvious that \( \hat{E}_0 = 0 \) since \( B_0 = A_0 \) in the multigrid algorithm. Assume the estimate (5.39) holds for \( j - 1 \). In the case of \( j \), we have from the identity (5.37) that
\[
\| \hat{E}_j \|^2_a \leq \| (I - P_{j-1})K_jv \|^2_a + \delta \| P_{j-1}K_jv \|^2_a
\]
\[
= (1 - \delta)\| (I - P_{j-1})K_jv \|^2_a + \delta(\| (I - P_{j-1})K_jv \|^2_a + \| P_{j-1}K_jv \|^2_a)
\]
\[
= (1 - \delta)\| (I - P_{j-1})K_jv \|^2_a + \delta\| K_jv \|^2_a
\]
\[
\leq (1 - \delta)\frac{C_1}{2C_2}(\|v\|^2_a - \|K_jv\|^2_a) + \delta\| K_jv \|^2_a = \delta\|v\|^2_a.
\]
This completes the proof.

Finally, since \( \|E_j\| = \|K_j\hat{E}_j\|_a \leq \|K_j\|_a \|\hat{E}_j\|_a \) and \( \|K_j\|_a \leq 1 \), we get the following estimate for the standard (V-cycle) multigrid algorithm,
\[
\|E_j\|^2_a \leq \frac{C_1}{C_1 + 2C_2} \tag{5.41}
\]
where the constants \( C_1 \) and \( C_2 \) both are independent of \( J \). The estimate (5.41) implies that, under the full regularity assumptions, the algorithm converges uniformly.
total work is linearly proportional to that involved in the smoothing steps in the finite element space $V_J = V$, which is on the order of the number of space dimensions of $V$, i.e., the number of unknowns in a discrete linear system.

It is noteworthy that, if the regularity assumptions are violated (say, the domain $\Omega$ has re-entrant corners), the estimate (5.41) does not hold and the constants $C_1$ and $C_2$ will depend on $j$. Fortunately, there exist techniques for proving the convergence rate of the multigrid algorithm without using regularity assumptions [42, 43, 195].

The convergence theory without regularity assumptions, originally proposed by Bramble and Pasciak et al. [43], is formulated in the abstract framework of the multiplicative subspace correction methods introduced in the previous section. It is based on the decomposition of the space $V$ into a direct sum

$$V = W_0 \oplus W_1 \oplus \cdots \oplus W_J \quad (5.42)$$

of subspaces $W_j \subseteq V$, where $\{V_j\}_{j=0}^J$ is a nested sequence of subspaces of $V$ as (5.20). In effect, each of the three space decomposition schemes discussed in the previous section may be applied to generate the direct sum. The subspaces $W_j$ are only a tool for the theoretical analysis. They do not enter the practical computation.

Two assumptions have to be fulfilled to apply the theory. The first assumption concerns the stability of the decomposition. We require that there exists a constant $C_3$ such that, for all $w_j \in W_j$,

$$\sum_{j=0}^{J} (R_j^{-1}w_j, w_j) \leq C_3 \| \sum_{j=0}^{J} w_j \|_a^2. \quad (5.43)$$

The second assumption is a Cauchy-Schwarz type inequality. We assume that there exist constants $\gamma_{kl} = \gamma_{lk}$ with

$$a(w_k, v_l) \leq \gamma_{kl} \left( R_k^{-1}w_k, w_k \right)^{1/2} \left( R_l^{-1}v_l, v_l \right)^{1/2} \quad (5.44)$$
for $k \leq l$, all $w_k \in V_k$ and all $v_l \in W_l$, such that

$$
\sum_{k,l=0}^{J} \gamma_{kl} x_k y_l \leq C_4 \left( \sum_{k=0}^{J} x_k^2 \right)^{1/2} \left( \sum_{l=0}^{J} y_l^2 \right)^{1/2}
(5.45)
$$

holds for all $x_k, y_l \in \mathcal{R}$. This means, we require that the spectral radius of the matrix $(\gamma_{kl})$ is bounded by a constant $C_4$.

In addition, we assume that there exists a positive constant

$$
0 < \omega < 2,
(5.46)
$$

such that

$$
(R_j v, v) \leq \omega (A_j^{-1} v, v) \quad \forall v \in V_j.
(5.47)
$$

or equivalently,

$$
(A_j v, v) \leq \omega (R_j^{-1} v, v) \quad \forall v \in V_j.
(5.48)
$$

This assumption is simply that all relaxations $R_j$ in the multigrid iteration are convergent.

Note that the inequalities (5.44) and (5.45) imply that

$$
\| \sum_{j=0}^{J} w_j \|_a^2 \leq C_4 \sum_{j=0}^{J} (R_j^{-1} w_j, w_j)
(5.49)
$$

for all $w_j \in W_j$. Therefore the expression

$$
\| \sum_{j=0}^{J} w_j \|_a^2 = \sum_{j=0}^{J} (R_j^{-1} w_j, w_j)
$$

defines a norm on $V$, which is equivalent to the energy norm induced by the bilinear form $a(\cdot, \cdot)$.  

141
Under the assumptions above, the standard V-cycle multigrid algorithm reduces the energy norm of the error at least by a factor $\|E_J\|_a$ where \[\|E_J\|_a^2 \leq 1 - \frac{2 - \omega}{C_3(1 + C_4)^2}. \tag{5.50}\]

However, there are situations in which the assumption on the Cauchy-Schwarz type inequality (5.44) causes problems, especially if the coefficients functions of the differential operator under consideration are not smooth or even not differentiable. In such cases, it is often still possible to prove the norm estimate (5.49), which is sufficient to derive an estimate for the norm of the error propagation error. It is known that, assuming only (5.49) instead of both (5.44) and (5.45), the norm of the error propagation operator satisfies the estimate \[\|E_J\|_a^2 \leq 1 - \frac{2 - \omega}{C_3(1 + \sqrt{\omega C_4})^2}. \tag{5.51}\]

This still shows that the convergence rate does not deteriorate too rapidly in terms of the number of subspaces.

In fact, in the application of our AMR algorithm to the singularly-perturbed reaction-diffusion problems, if the coefficients and the data in the monodomain or the bidomain models are just good enough (not so good that the problems are $H^2$-regular), the assumption (5.49) can be easily satisfied.

Note that the adaptively refined grids in our AMR algorithm are locally quasi-uniform if the given coarsest grid $G_0$ in the hierarchy is shape-regular [34, 57]. The adaptive mesh refinement preserves the local quasi-uniformity of grids, due to the proper nesting of levels and the uniform refinement of tagged subgrids. It is proved by Bramble et al. [40] that, if the coarsest grid $G_0$ in the AMR hierarchy is appropriately selected, the local quasi-uniformity of composite grids guarantees that the
$L^2$-orthogonal projections $Q_j$ onto their corresponding finite element spaces $V_j$ are stable in $H^1(\Omega)$, i.e., there is a constant $C$ independent of $j$ such that

$$\|Q_j v\|_{H^1(\Omega)} \leq C \|v\|_{H^1(\Omega)} \quad (5.52)$$

for any $v \in H^1(\Omega)$. For the variational problem (5.1), the stability condition (5.52) is equivalent to

$$\|Q_j v\|_a \leq C \|v\|_a \quad (5.53)$$

for any $v \in H^1(\Omega)$.

**Remark 5.3:** The proof for the stability of the $L^2$ projection in $H^1(\Omega)$ is based on the assumption that the volume of neighboring elements in a locally quasiuniform and geometrically refined mesh does not change too drastically [40]. The condition may be violated on the composite grids generated by our algorithm. Especially, when the mesh refinement ratio is relatively large ($r = 4$ or $8$) in our AMR algorithm, any two neighboring cells across a coarse fine interface boundary will have a large ratio of volumes, which may bring trouble. In this case, quasi-interpolation or $L^2$-like (local) projections [58, 64] should be used instead. Nevertheless, for the simplicity of exposition, we still assume in rest of this section that the $L^2$ orthogonal projections $Q_j$ are stable in $H^1(\Omega)$.

The $L^2$-orthogonal projections $Q_j$ give us a tool to perform a stable decomposition of the space $V$. Actually, the splitting (5.18) uses the $L^2$-orthogonal projections $Q_j$ to generate such a decomposition. That is, any $v \in V$ can be represented as a sum of elements $w_j$ from the subspaces $W_j$, i.e.,

$$v = w_0 + w_1 + \cdots + w_J, \quad (5.54)$$

where $w_j \equiv (Q_j - Q_{j-1}) v$ and $Q_J = I$, $Q_{-1} = 0$. 

143
The approximation property (5.33) implies that
\[
(R^{-1}_j w_j, w_j) \lesssim \rho(A_j)(w_j, w_j) = \rho(A_j)\|(Q_j - Q_{j-1})v\|^2, \tag{5.55}
\]
for \(w_j = (Q_j - Q_{j-1})v \in W_j\). Here, \(A \lesssim B\) means that there is a generic constant \(c\) independent of the number of refinement levels \(j\) such that \(A \leq cB\).

Considering that the \(L^2\)-orthogonal projection \(Q_j\) satisfies the error estimate
\[
\|v - Q_j v\|^2 \lesssim \frac{1}{\rho(A_j)}\|v\|^2_a, 
\]
we obtain an upper bound for the right hand side of (5.55)
\[
\rho(A_j)\|(Q_j - Q_{j-1})v\|^2 \lesssim \frac{\rho(A_j)}{\rho(A_{j-1})}\|Q_j v\|^2_a \lesssim \|v\|^2_a. \tag{5.56}
\]
Here, the stability of the \(L^2\) projection \(Q_j\) is used.

Combining the inequalities (5.56) and (5.55) yields
\[
\sum_{j=0}^{J} (R^{-1}_j w_j, w_j) \lesssim (J + 1)\|v\|^2_a = (J + 1) \sum_{j=0}^{J} \|w_j\|^2_a. \tag{5.57}
\]
This means that the decomposition of space by the \(L^2\) projections is stable.

On the other hand, the smoothing property (5.34) implies that
\[
\|w_j\|^2_a = (A_j w_j, w_j) \lesssim (R^{-1}_j w_j, w_j),
\]
for \(w_j = (Q_j - Q_{j-1})v \in W_j\). Then, we obtain the following inequality
\[
\|\sum_{j=0}^{J} w_j\|^2_a \leq (J + 1)\sum_{j=0}^{J} \|w_j\|^2_a \lesssim (J + 1)\sum_{j=0}^{J} (R^{-1}_j w_j, w_j). \tag{5.58}
\]

Hence, the two assumptions (5.43) and (5.49) are satisfied respectively by (5.57) and (5.58) except that the constants \(C_3\) and \(C_4\) are dependent of the number of
subspaces (levels). In effect, by the equivalence of certain Besov and Sobolev spaces, the upper and lower bounds estimate (5.57) and (5.58) can be improved to

\[ \sum_{j=0}^{J} (R_j^{-1}w_j, w_j) \leq \tilde{C}_3 \left\| \sum_{j=0}^{J} w_j \right\|^2, \]

\[ \left\| \sum_{j=0}^{J} w_j \right\|^2 \leq \tilde{C}_4 \sum_{j=0}^{J} (R_j^{-1}w_j, w_j), \]

where \(\tilde{C}_3\) and \(\tilde{C}_4\) neither depend on the number of subspaces (levels) nor on regularity properties of the variational problem to be solved [32, 140].

Finally, we have to point out that the estimates (5.50) and (5.51) work for general multiplicative subspace correction methods, including domain decomposition methods. It is natural that the estimates apply to multigrid algorithms, which are special multiplicative methods.

As a matter of fact, for the standard \(V\)-cycle multigrid algorithm, Bramble et al. proved the following convergence estimate without regularity assumptions [43]. Assume that there are linear mappings \(\Pi_j : V \to V_j, \Pi_J = I\) (not necessarily projectors), and positive constants \(c_1\) and \(c_2\) such that

\[ \| \Pi_j v \|_a \leq c_1 \| v \|_a, \]

\[ \| (\Pi_j - \Pi_{j-1})v \| \leq \frac{c_2}{\rho(A_j)} \| v \|_a, \]

for any \(v \in V\). Assume that the smoothers \(S_j\) satisfy the following approximation property with positive constant \(C_s\):

\[ \frac{\| u \|^2}{\rho(A_j)} \leq C_s (S_j u, u), \]

for all \(u \in \tilde{V}_j\), where \(\tilde{V}_j\) is the range of \(S_j\), and \(\tilde{V}_j \supset Range(\Pi_j - \Pi_{j-1})\). Then, the
standard $V$-cycle multigrid algorithm admits the following convergence estimate:

$$\|E_J\|_a = \|I - B_J^{-1}A_J\|_a \leq 1 - \frac{C}{J}$$

with $C = 1/(1+c_1+c_2 C_1^{1/2})^2$. The existence of the linear mappings $\Pi_j$ for the linear diffusion problems on locally refined grids has been verified by Trangenstein [181].

**Remark 5.4**: It is noteworthy that, in an adaptive mesh refinement application, the range $\tilde{V}_j$ of the smoother $S_j$ only contains degrees of freedom in a fine subdomain and hence smoothing need only be done in that subdomain.

In the thesis work, we will employ the standard $V$-cycle multigrid algorithm to solve linear systems resulting from discretization of diffusion on uniformly or locally refined (composite) grids. By the convergence rate estimate (5.61), where the upper bound is always less than one and positive, given that the maximum number of refinement levels in our algorithm is fixed, the $V$-cycle multigrid method on uniformly or locally refined (composite) grids converges within a fixed number of iterations to a specified precision.

### 5.4 Iteration on Uniformly Refined Grids

So far, we have discussed the standard $V$-cycle multigrid algorithm in an abstract formulation. In the next, we will describe some details of the algorithm in a more specific way from the viewpoint of implementation.

Recall that our AMR algorithm uses a hierarchy of nested levels $\ell_j$ ($j = 0, \cdots, J$). Each level $\ell_j$ has a main grid $G_j$, which results from uniform refinement of the tagged subgrid $\hat{G}_{j-1}$ of $G_{j-1}$ on the parent level $\ell_{j-1}$ (see Figure 4.7). Let $\Omega_j$ be the region covered by the main grid $G_j$ on level $\ell_j$. It is obvious that the subdomains $\Omega_j$ are
properly nested too

\[ \Omega_0 \supseteq \Omega_1 \supseteq \cdots \supseteq \Omega_J. \]

The main grid \( G_j \) makes up a partition \( T_j \) of the subdomain \( \Omega_j \) with mesh parameter \( h_j \), i.e., \( T_j \equiv G_j \). Each partition \( T_j \) may have two different kinds of boundary nodes. Some are on the physical boundary \( \partial \Omega \) of the global domain \( \Omega \). Others are on the interface boundary \( \Gamma_{j-1/2} \) between the subdomains \( \Omega_{j-1} \) and \( \Omega_j \). Based on the partition \( T_j \) of the subdomain \( \Omega_j \) on each level \( \ell_j \), a finite element space \( W_j \) is defined by

\[
W_j = \{ \chi \in C(\bar{\Omega}_j) : \chi|_T \cdot J(\xi) \in Q_1(\hat{T}), \ \forall T \in T_j, \ \text{and} \ \chi = 0 \ \text{on} \ \Gamma_{j-1/2} \}, \quad (5.62)
\]

where \( Q_1(\hat{T}) \) is the space of the bilinear (2D) or trilinear (3D) functions on the reference element \( \hat{T} \), and \( J(\xi) \) is the bijective mapping defined in Figure 3.3 (2D) or Figure 3.4 (3D). The finite element space \( W_j \) consists of the continuous piecewise linear functions that vanish on the interface boundary \( \Gamma_{j-1/2} \).

In this section, we only consider the case that the grids are uniformly or globally refined. This means, all subdomains \( \Omega_j \) are identical to the global domain \( \Omega \). The sequence of finite element spaces \( W_j \) is automatically nested, i.e.,

\[ W_0 \subset W_1 \subset \cdots \subset W_J. \]

Let \( V_j \equiv W_j \) for \( j = 0, 1, \cdots, J \). The standard \( V \)-cycle multigrid algorithm can then be applied for solving finite element equations like (3.55) or

\[
A_j u_j = b_j, \quad (5.63)
\]

where \( A_j \) is a symmetric and positive definite matrix, and \( u_j, b_j \) are vectors.

### 5.4.1 Smoothing/Relaxation

In the case of uniform refinement, it is trivial to implement the pre-smoothing and the post-smoothing steps in the multigrid algorithm. As stated, any convergent
iteration method, such as the Richardson iteration, the weighted Jacobi iteration, the Gauss-Seidel iteration or the SOR scheme, can be used as a smoother \( S_j \). Note that sometimes (e.g., the grid \( G_j \) has a large aspect ratio) it is necessary to scale or damp smoothers \( S_j \) such that the relaxation operators \( R_j \) satisfy the approximation property (5.33).

The smoothing procedure updates a solution vector one by one component, following the order that the nodes appear in the grid representation, if the data is node-centered. That is, by the hierarchical ordering of grid entities, the data associated with the boundary nodes are first updated and then those associated with the interior nodes; the data associated with the nodes from refinement of those belonging to the same kind of grid entity are updated in a row. For example, in three space dimensions, the data associated with the old boundary nodes are updated first, followed by the data associated with the new nodes from refinement of the old boundary edges, next the data associated with the new nodes from refinement of the old boundary sides, then the data associated with the old interior nodes, the data associated with the new nodes from refinement of the old interior edges, those associated with the new nodes from refinement of the old interior sides and finally those associated with the new nodes from refinement of the old cells (see Figures 4.5-4.6). Here, the “old” nodes are those appearing on both the coarse and the fine grids. The “new” nodes are those only appearing on the fine grid but not on the coarse grid.

As noted, if the multigrid iteration is used as a preconditioner for another iteration procedure such as a conjugate gradient, the smoothers in the pre-smoothing and the post-smoothing steps must be adjoint to each other or symmetric. A symmetrization technique is to update (smooth) the data again after its first smoothing, but in a reverse order. In our current implementation of the AMR algorithm, we have not used the conjugate gradient iteration yet.
**Subspace Correction** on level $\ell_j$:

1. Compute the fine residual $r_j = g_j - A_j u_j^\nu$.
2. Restrict the residual to the coarse grid $g_{j-1} = R_{j-1} r_j$.
3. Solve the coarse grid equations $v_{j-1} = B_{j-1} g_{j-1}$.
4. Prolong the coarse grid correction $c_j = P_{j-1} v_{j-1}$.
5. Correct the solution $u_j^{\nu+1} = u_j^\nu + c_j$.

Figure 5.1: Subspace correction on Level $\ell_j$ (uniformly refined grids)

### 5.4.2 Coarse Grid Correction

As we know, another component of a multigrid iteration is the subspace correction (5.25). It involves the $L^2$-orthogonal projection $Q_{j-1}$. To correctly implement the projection $Q_{j-1}$, let us first introduce two important operators: the prolongation operator $P_{j-1}$ and the restriction operator $R_{j-1}$. It is natural that the prolongation operator $P_{j-1}$ is defined by the linear interpolant $P_{j-1} \equiv I_{j-1} : V_{j-1} \to V_j$ since we are working with piecewise linear finite elements. The prolongation $P_{j-1}$ is an injection of the coarse space $V_{j-1}$ into the fine space $V_j$. The restriction operator $R_{j-1}$ is defined by the adjoint with respect to the $L^2$ inner product of the prolongation $R_{j-1} = I_{j-1}^\top : V_j \to V_{j-1}$. Denote by $(\cdot, \cdot)_j$ the $L^2$ inner product in the space $V_j$.

By the definition of $Q_{j-1}$ and the linearity of elements in $V_{j-1}$ and $V_j$, we have

$$(Q_{j-1} u_j, v_{j-1})_{j-1} = (u_j, v_{j-1})_j = (u_j, I_{j-1}^\top v_{j-1})_j = (I_{j-1}^\top u_j, v_{j-1})_{j-1},$$

for all $u_j \in V_j$ and $v_{j-1} \in V_{j-1}$. It is obvious that the restriction operator $R_{j-1}$ is exactly identical to the $L^2$-orthogonal projection $Q_{j-1}$.

With the prolongation operator $P_{j-1}$ and the restriction operator $R_{j-1}$, the subspace correction step (5.25) is equivalent to the following iteration in terms of matrices.
and vectors
\[ u_{j}^{\nu+1} = u_{j}^{\nu} + P_{j-1}^j B_{j-1}^{-1} R_{j}^{j-1}(g_{j} - A_{j} u_{j}^{\nu}). \] (5.64)

Here, \( u_{j}^{\nu} \) is the vector of nodal values of the function \( u_{j}^{\nu} \in V_{j} \); the vector \( g_{j} \) also corresponds to an element in \( V_{j} \); \( P_{j-1}^j \) and \( R_{j}^{j-1} \) are understood as linear operators on vectors as well as on elements of finite element subspaces. For a subspace correction on level \( \ell_{j} \), see Figure 5.1.

In our AMR algorithm, the implementation of the prolongation and restriction operators is essentially the same as up-/downscaling in section 4.4.5.

### 5.5 Iteration on Locally Refined Grids

In this section, we consider the general case that the grids are locally refined. In this case, the finite element spaces \( W_j \) defined in the previous section are not nested. However, a nested sequence of finite element spaces \( V_j \) can be created by

\[ V_j \equiv \bigcup_{l=0}^{j} W_l \quad \text{for } j = 0, 1, \ldots, J. \] (5.65)

Let \( V \) be the largest space \( V_J \). Then all other spaces \( V_j \) are subspaces of \( V \). It is obvious that each of the subspaces \( V_j \) is made up of continuous piecewise linear functions. This implies the finite element spaces \( V_j \) are subspaces of the Sobolev space \( H^1(\Omega) \), i.e., \( V_j \subset H^1(\Omega) \). In this sense, we say the finite element approximation is conforming.

Let \( G_j \) be the composite grid consisting of all of the untagged cells on the coarse levels \( \ell_k \) \( (k = 0, 1, \ldots, j - 1) \) and the main grid cells of \( G_j \) on the fine level \( \ell_j \), i.e.,

\[ G_j \equiv G_j \bigcup_{k=0}^{j-1} \left( \bigcup_{k=0}^{i-1} G_k \setminus \hat{G}_k \right). \]
Figure 5.2: A two-level composite domain

It is easy to find that, for each $j > 0$, the finite element subspace $V_j$ exactly consists of the functions that are continuous and piecewise linear on the composite grid $G_j$. As a matter of fact, all of the subspaces $V_j$ can be used to discretize a general variational problem such as (3.53) from implicit time integration of the linear diffusion equations (3.45). After the discretizations, we will obtain linear systems in the same form as (5.63). Then it is straightforward to apply the standard $V$-cycle multigrid algorithm to solve for the solutions. This approach is classic.

However, in the thesis work, we formulate the variational problems on locally refined grids in a different way even though our approach is essentially equivalent to the classic one. As illustrated by (3.60a)-(3.60b) and (3.61a)-(3.61b), the variational problem on a composite grid is presented as an interface problem in our approach: two sub-problems on the coarse and the fine subdomains are formed independently and coupled through the interface boundary by the Steklov-Poincare continuity conditions, which are weakly enforced using Lagrange multipliers. The composite grid equations resulting from our approach will finally have the same form as (5.63) if the degrees of freedom associated with the hanging nodes in a composite grid are
correctly eliminated (condensed) [181].

In our AMR algorithm, a composite grid iteration proceeds recursively on a grid hierarchy and works with at most two grids in any region of space. For this reason, we restrict ourselves to discuss only the case that our computational domain, denoted by \( \Omega_{j-1} \), is composed of a coarse subdomain, denoted by \( \hat{\Omega}_{j-1}^* \) and a fine subdomain, denoted by \( \Omega_j \); see Figure 5.2(a). Assume that \( \Omega_j \cap \hat{\Omega}_{j-1}^* = \emptyset \) and \( \Omega_{j-1} = \hat{\Omega}_{j-1}^* \cup \hat{\Omega}_j \).

Let \( \Gamma_{j-1/2} = (\partial \Omega_j \cap \partial \hat{\Omega}_{j-1}^*) \setminus \partial \Omega_{j-1} \neq \emptyset \) be the interface between the coarse and the fine subdomains. Let \( \hat{T}_{j-1}^* \) and \( T_j \) be two regular and conformal partitions of the coarse subdomain \( \hat{\Omega}_{j-1}^* \) and the fine subdomain \( \Omega_j \), respectively; see Figure 5.2(b). We assume that the fine partition \( T_j \) results from uniform refinement of another coarse partition \( \hat{T}_{j-1} \) of the subdomain \( \Omega_j \). We also assume that the union of the two coarse partitions \( \hat{T}_{j-1} \) and \( \hat{T}_{j-1}^* \) makes up a conformal partition of the computational domain \( \Omega_{j-1} \).

In fact, in our AMR algorithm, the fine subdomain \( \Omega_j \) corresponds to the region covered by the main grid \( G_j \) on level \( \ell_j \). The coarse subdomain \( \hat{\Omega}_{j-1}^* \) corresponds to the region covered by the untagged cells on level \( \ell_{j-1} \) and it also represents the complement of the subdomain \( \hat{\Omega}_{j-1} \) that are covered by the tagged subgrid \( \hat{G}_{j-1} \); see Figure 4.7.

Discretizing a second-order self-adjoint linear elliptic boundary value problem (or a linear diffusion problem like (3.45)) on the two-level domain yields a system of composite grid equations like (3.71)-(3.72), which are in the following form

\[
A_j \mathbf{u}_j - \mathbf{m}_j = \mathbf{b}_j \quad \text{in } \Omega_j, \tag{5.66a}
\]

\[
\hat{A}_{j-1}^* \hat{\mathbf{u}}_{j-1} + \hat{\mathbf{m}}_{j-1}^* = \hat{\mathbf{b}}_{j-1}^* \quad \text{in } \hat{\Omega}_{j-1}^*, \tag{5.66b}
\]
(a) $\mathcal{R}^{-1}_j \mathbf{u}_{j-1}^{(b)} = \frac{1}{2} \mathbf{u}_j^{(d)} + \mathbf{u}_j^{(b)} + \frac{1}{2} \mathbf{u}_j^{(e)}$

(b) $\mathcal{R}^{-1}_j \mathbf{u}_{j-1}^{(b)} = \frac{1}{2} \mathbf{u}_j^{(d)} + \mathbf{u}_j^{(b)} + \frac{1}{2} \mathbf{u}_j^{(e)}$

(c) $\mathcal{R}^{-1}_j \mathbf{u}_{j-1}^{(e)} = \mathbf{u}_j^{(e)} + \frac{1}{2} (\mathbf{u}_j^{(f)} + \mathbf{u}_j^{(g)} + \mathbf{u}_j^{(h)} + \mathbf{u}_j^{(i)})$

(d) $\mathcal{R}^{-1}_j \mathbf{u}_{j-1}^{(b)} = \frac{1}{2} \mathbf{u}_j^{(d)} + \mathbf{u}_j^{(b)} + \frac{1}{2} \mathbf{u}_j^{(e)}$

**Figure 5.3:** Interface restriction $\mathcal{R}^{-1}_j \equiv \mathcal{I}^{-1}_j$ from level $\ell_j$ to level $\ell_{j-1}$
subject to the constraints along the interface boundary $\Gamma_{j-1/2}$

$$
\hat{m}_{j-1}^* = \mathcal{R}_{j-1}^j m_j \quad \text{on } \Gamma_{j-1/2}, \quad (5.67a)
$$
$$
u_j = \mathcal{P}_{j-1}^j \hat{u}_{j-1}^* \quad \text{on } \Gamma_{j-1/2}. \quad (5.67b)
$$

Here, $A_j$, $\hat{A}_{j-1}^*$ are symmetric and positive definite matrices. $u_j$, $\hat{u}_{j-1}^*$, $m_j$, $\hat{m}_{j-1}^*$, $b_j$ and $\hat{b}_{j-1}^*$ are vectors. The matrix $A_j$ and the vectors $u_j$, $m_j$, $b_j$ correspond to the discretization on the fine subdomain $\Omega_j$; and the matrix $\hat{A}_{j-1}^*$ and the vectors $\hat{u}_{j-1}^*$, $\hat{m}_{j-1}^*$, $\hat{b}_{j-1}^*$ correspond to the discretization on the coarse subdomain $\hat{\Omega}_{j-1}^*$, i.e.,

$$A_j, u_j, m_j, b_j \sim \Omega_j, \quad \text{and} \quad \hat{A}_{j-1}^*, \hat{u}_{j-1}^*, \hat{m}_{j-1}^*, \hat{b}_{j-1}^* \sim \hat{\Omega}_{j-1}^*.$$

The symbol $\mathcal{P}_{j-1}^j$ represents the interface prolongation operator, which is defined by the interface interpolant $I_{j-1}^j : \Gamma_{j-1/2}(\Omega_{j-1}) \to \Gamma_{j-1/2}(\Omega_j)$ (from the coarse domain side to the fine domain side). The symbol $\mathcal{R}_{j-1}^j$ denotes the interface restriction operator, which is defined as the adjoint of the interface prolongation operator, i.e., $\mathcal{R}_{j-1}^j \equiv I_{j-1}^j$; see Figure 5.3.

5.5.1 Smoothing/Relaxation

In order to perform the pre-smoothing and the post-smoothing steps in the multigrid algorithm, we always fix the fine residual $r_j$ on the interface boundary $\Gamma_{j-1/2}$ to be zero, i.e.,

$$r_j = b_j - A_j u_j + m_j \equiv 0 \quad \text{on } \Gamma_{j-1/2} \quad (5.68)$$

since the fine subdomain problem on the interface boundary $\Gamma_{j-1/2}$ has the Dirichlet boundary conditions that are interpolated from the coarse subdomain side. The identity (5.68) implies that

$$m_j = A_j u_j - b_j \quad \text{on } \Gamma_{j-1/2}. \quad (5.69)$$
By the interface condition (5.67a), we get
\[ \hat{m}_{j-1}^* = \mathcal{R}^{-1}_j(A_j u_j - b_j) \quad \text{on } \Gamma_{j-1/2}. \] (5.70)

On the other hand, the coarse residual in the coarse subdomain \( \hat{\Omega}_{j-1}^* \) is given by
\[ \hat{r}_{j-1}^* = (\hat{b}_{j-1}^* - \hat{A}_{j-1}^* \hat{u}_{j-1}) - \hat{m}_{j-1}^* \quad \text{on } \hat{\Omega}_{j-1}^* \cup \Gamma_{j-1/2}. \] (5.71)

Substituting (5.70) into (5.71) yields
\[ \hat{r}_{j-1}^* = (\hat{b}_{j-1}^* - \hat{A}_{j-1}^* \hat{u}_{j-1}) + \mathcal{R}_j^{-1}(b_j - A_j u_j) \quad \text{on } \hat{\Omega}_{j-1}^* \cup \Gamma_{j-1/2}. \] (5.72)

After the coarse residual is computed, the data in the coarse subdomain \( \hat{\Omega}_{j-1}^* \) can be updated by
\[ \hat{u}_{j-1}^* \leftarrow \hat{u}_{j-1}^* + S_{j-1} \hat{r}_{j-1}^* \quad \text{on } \hat{\Omega}_{j-1}^* \cup \Gamma_{j-1/2}, \] (5.73)

where \( S_{j-1} \) is the smoother on level \( \ell_{j-1} \). Next by the interface condition (5.67b), the interface data on the fine subdomain side is interpolated by
\[ u_j \leftarrow \mathcal{P}^j_{j-1} \hat{u}_{j-1}^* \quad \text{on } \Gamma_{j-1/2}. \] (5.74)

Note that the fine residual on the interior nodes of \( \Omega_j \) is given by
\[ r_j = b_j - A_j u_j \quad \text{in } \Omega_j \setminus \Gamma_{j-1/2}. \] (5.75)

Finally, the fine data on the interior nodes of the subdomain \( \Omega_j \) is updated by
\[ u_j \leftarrow u_j + S_j r_j \quad \text{in } \Omega_j \setminus \Gamma_{j-1/2}, \] (5.76)

where \( S_j \) is the smoother on level \( \ell_j \). The procedure above from (5.68) to (5.76) defines a smoothing step in the multigrid algorithm. For an example of the smoothing for composite grid equations, see Figure 5.4, where the order of the steps is slightly different from above but it should not hurt the convergence of the smoothing. Note that the iteration method used for the interface data relaxation (e.g., Jacobi) may be different from that used for the interior data relaxation (e.g., Gauss-Seidel) in order to reduce the cost of data communication between the coarse and the fine subdomains in the case of distributed computing.
**Smoothing/relaxation** on the composite grid $G_j$:

1. Residual computation (except those on hanging nodes)

$$r_j \leftarrow g_j - A_j u_j \quad \text{on } \Omega_j \cup \Gamma_{j-1/2},$$
$$\hat{r}^*_j \leftarrow \hat{b}^*_j - \hat{A}^*_j \hat{u}^*_{j-1} \quad \text{on } \hat{\Omega}^*_j \cup \Gamma_{j-1/2},$$

where $g_j \equiv b_j$ only on the finest level $\ell_j$.

2. Interior data relaxation (e.g., Gauss-Seidel iteration)

$$u_j \leftarrow u_j + S_j r_j \quad \text{in } \Omega_j,$$
$$\hat{u}^*_{j-1} \leftarrow \hat{u}^*_{j-1} + S_{j-1} \hat{r}^*_{j-1} \quad \text{in } \hat{\Omega}^*_{j-1}.$$

3. Interface residual restriction (from $\Omega_j$ to $\hat{\Omega}^*_j$)

$$\hat{r}^*_{j-1} \leftarrow \hat{r}^*_{j-1} + \mathcal{R}^{j-1}_j r_j \quad \text{on } \Gamma_{j-1/2}.$$

4. Interface data relaxation (e.g., Jacobi iteration)

$$\hat{u}^*_{j-1} \leftarrow \hat{u}^*_{j-1} + \hat{S}^*_{j-1} \hat{r}^*_{j-1} \quad \text{on } \Gamma_{j-1/2}.$$

5. Interface data prolongation (from $\hat{\Omega}^*_j$ to $\Omega_j$)

$$u_j \leftarrow \mathcal{P}^j_{j-1} \hat{u}^*_{j-1} \quad \text{on } \Gamma_{j-1/2}.$$

**Figure 5.4:** Smoothing/relaxation for composite grid equations
5.5.2 Coarse Grid Correction

The coarse grid correction on locally refined grids is essentially the same as that on uniformly refined grids. It also consists of the fine residual computation, the fine residual restriction, the recursive invocation of the multigrid algorithm, the coarse data prolongation and the fine data correction.

First, note that the right hand side $b_j$ in the composite grid equation (5.66a) should be replaced by $g_j$, the input data of the multigrid algorithm on the current level $\ell_j$. The data $g_j$ equals $b_j$ only if it is on the finest level $\ell_J$. After the replacement, we can compute residuals by the composite grid equations (5.66)

$$r_j \leftarrow g_j - A_j u_j \quad \text{on } \Omega_j \cup \Gamma_{j-1/2}, \quad (5.79a)$$

$$\hat{r}_{j-1}^* \leftarrow \hat{b}_{j-1}^* - \hat{A}_{j-1}^* \hat{u}_{j-1}^* \quad \text{on } \hat{\Omega}_{j-1} \cup \Gamma_{j-1/2}. \quad (5.79b)$$

Second, the fine residual $r_j$ on the fine level $\ell_j$ is restricted to the coarse level $\ell_{j-1}$ using the intergrid operator $R_{j}^{j-1} \equiv I_{j}^{j-1}$. In the AMR algorithm, the fine residual restriction is implemented through the correspondence between the fine grid $G_j$ and the tagged subgrid $\hat{G}_{j-1}$. For this reason, we denote by $\hat{r}_{j-1}$ the restricted residual, which corresponds to the interior nodes of the tagged subgrid $\hat{G}_{j-1}$. That is,

$$\hat{r}_{j-1} \leftarrow R_{j}^{j-1} r_j. \quad \text{from } \Omega_j \text{ to } \Omega_{j-1}. \quad (5.80)$$

On the other hand, similar to those in the smoothing steps, the interface component of the coarse residual $\hat{r}_{j-1}^*$, which corresponds to the nodes on the interface boundary $\Gamma_{j-1/2}$, must be corrected by the restricted data, i.e.,

$$\hat{r}_{j-1}^* \leftarrow \hat{r}_{j-1}^* + R_{j}^{j-1} r_j \quad \text{on } \Gamma_{j-1/2}. \quad (5.81)$$

Now, the two components $\hat{r}_{j-1}$ and $\hat{r}_{j-1}^*$ make up a residual vector $r_{j-1}$ on the coarse level $\ell_{j-1}$: $r_{j-1} \equiv (\hat{r}_{j-1}; \hat{r}_{j-1}^*)^T$. 

157
In the third step, we set $g_{j-1} = r_{j-1}$ and recursively invoke the $V$-cycle multigrid algorithm on the coarse level $\ell_{j-1}$, where $g_{j-1}$ is the input data. After solving the composite grid equations on $G_{j-1}$, we will get a solution vector $v_{j-1}$ with two components, $\hat{v}_{j-1}$ and $\hat{v}^*_{j-1}$, corresponding to the tagged subgrid $\hat{G}_{j-1}$ and its complement $\hat{G}^*_{j-1}$. That is,
\[
 v_{j-1} \leftarrow B^{-1}_{j-1} g_{j-1} \quad \text{on } G_{j-1}, \tag{5.82}
\]
where $v_{j-1} \equiv (\hat{v}_{j-1}, \hat{v}^*_{j-1})^T$.

Fourth, from the vector $v_{j-1}$, we extract out the component $\hat{v}_{j-1}$ that corresponds to the tagged subgrid $\hat{G}_{j-1}$ and prolong it to the fine level $\ell_j$ by
\[
 c_j \leftarrow \mathcal{P}_{j-1}^j \hat{v}_{j-1} \quad \text{from } \Omega_{j-1} \text{ to } \Omega_j. \tag{5.83}
\]

In the last step, the interpolated data $c_j$ is used as a correction for the approximate solution $u_j$ of the composite grid equations on $G_j$:
\[
 u_j \leftarrow u_j + c_j \quad \text{in } \Omega_j. \tag{5.84}
\]

The five steps above compose a subspace correction in the $V$-cycle multigrid algorithm (see Figure 5.5).

### 5.6 Implementation Details

After the smoothing and the subspace correction steps are correctly implemented, the standard $V$-cycle multigrid algorithm is almost ready for solving linear equations except a few points.

**5.6.1 Initial Guess**

An initial guess for the solution of linear equations must be provided for the algorithm to start. Usually we assume that the initial guess is zero, especially in the case that the algorithm is used as a preconditioner for the conjugate gradient iteration.
Subspace Correction on the composite grid $G_j$:

1. Compute the composite grid residual

\[ r_j \leftarrow g_j - A_j u_j, \]
\[ \tilde{r}^*_j \leftarrow \tilde{b}^*_j - \tilde{A}^*_j \tilde{u}^*_j, \]

where $g_j \equiv b_j$ only on the finest level $\ell_J$.

2. Restrict the residual to level $\ell_{j-1}$

\[ \tilde{r}_{j-1} \leftarrow \mathcal{R}_{j-1} r_j, \]
\[ \tilde{r}^*_{j-1} \leftarrow \tilde{r}^*_{j-1} + \mathcal{R}_{j-1} r_j \]

and set $g_{j-1} = r_{j-1} \equiv (\tilde{r}_{j-1}, \tilde{r}^*_{j-1})^T$.

3. Solve the composite grid equations on $G_{j-1}$

\[ v_{j-1} \leftarrow B_{j-1}^{-1} g_{j-1} \]

where $v_{j-1} \equiv (\tilde{v}_{j-1}, \tilde{v}^*_{j-1})^T$.

4. Prolong the coarse grid correction

\[ c_j \leftarrow \mathcal{P}_{j-1} \tilde{v}_{j-1}. \]

5. Correct the fine data solution

\[ u_j \leftarrow u_j + c_j. \]

Figure 5.5: Subspace correction on level $\ell_j$ (locally refined grids)
For the equations resulting from the Crank-Nicolson finite element discretization of a linear diffusion problem such as (3.45), which is an initial and boundary value problem, the initial conditions usually provide a good initial guess for the multigrid iteration. When the time step length $\Delta t = t^{n+1} - t^n$ in the temporal discretization of the linear diffusion is small enough, the initial conditions at the old time $t^n$ yield an accurate approximation of the solution at the new time $t^{n+1}$. Such an initial guess will hopefully reduce the number of multigrid iterations.

5.6.2 Coarsest Level Solver

Recall that in the $V$-cycle multigrid algorithm, we assume that the multigrid operator $B_0$ on the coarsest level $\ell_0$, is identical to the Ritz approximation $A_0$ itself. This means that, the linear equations on the coarsest level $\ell_0$, which have the form

$$A_0 u_0 = g_0,$$

must be solved exactly, or at least very accurately. By our assumption on the root level grid (see section 5.1), this linear system is relatively small and can be solved effectively by the conjugate gradient iteration in multiple space dimensions, or by a direct solver in one space dimension.

In our current implementation of the AMR algorithm, a symmetric successive over-relaxation (SSOR) preconditioned conjugate gradient method is employed to solve the coarsest grid equations (5.87) since the coefficient matrix $A_0$ there is symmetric and positive definite.

5.6.3 Stopping Criterion

To achieve a finite precision, the algorithm has to be terminated within a finite number of iterations based on some stopping criterion.
First, it is well-known that, given any approximate solution \( \hat{u} \) to the linear system \( Au = b \), the relative error satisfies

\[
\frac{\|u - u^\nu\|}{\|u\|} \leq \kappa(A) \frac{\|b - Au^\nu\|}{\|b\|}.
\]

(5.88)

Here, \( \kappa(A) = \|A\|\|A^{-1}\| \) is the condition number of the invertible matrix \( A \). This estimate (5.88) implies that, provided that the condition number \( \kappa(A) \) is reasonable, we may conclude that the relative error is small if the residual is small relative to the right hand side.

In the thesis work, the matrix \( A \), resulting from the Crank-Nicolson finite element discretization of the linear diffusion (3.45), has the following form

\[
A = M + \tau K,
\]

where \( \tau \) is the temporal parameter; \( M \) and \( K \) are the mass and the stiffness matrices, respectively. We assume that the corresponding partition (composite grid) for the discretization satisfies the minimum angle property [6, 57]. Denote by \( h \) the minimum of the diameters of the largest element-inscribed balls in the partition (grid). By the Poincare inequality and an inverse estimate of the finite element functions, we have the following inequalities

\[
(1 + \epsilon \tau) (v, v) \lesssim (Av, v) = (Mv, v) + \tau (Kv, v) \lesssim (1 + \frac{\epsilon \tau}{h^2}) (v, v)
\]

(5.89)

for any \( v \) in the corresponding finite element space \( V \). Here and in the rest, \( A \lesssim B \) means that there is a generic constant \( c \) independent of the mesh parameters \( (h, \ell) \) or the temporal parameter \( \tau \) or the number of refinement levels \( J \) such that \( A \leq cB \). The constant \( \epsilon \) in the inequalities (5.89) represents the scale of the diffusion operator, whose discretization yields the stiffness matrix \( K \), relative to the Laplacian operator. For the bidomain model (3.1), the constant \( \epsilon \) is equal to the size of the
intra-/extracellular conductivity tensors $D_i$ and $D_e$. From those two inequalities in (5.89), we get an estimate of the condition number $\kappa(A)$

$$\kappa(A) \lesssim \frac{1 + \epsilon \tau/\hbar^2}{1 + \epsilon \tau} \lesssim \max\left\{1, \frac{\epsilon \tau}{\hbar^2}\right\}. \quad (5.90)$$

That is, the condition number of the matrix $\kappa(A)$ is on the order of $\epsilon \tau/\hbar^2$ if the ratio is greater than one, or $\kappa(A)$ is on the order of one if the ratio is small. Finally, from (5.88), the following relative error estimate is derived

$$\frac{\|u - u^\nu\|}{\|u\|} \lesssim \max\left\{1, \frac{\epsilon \tau}{\hbar^2}\right\} \frac{\|b - Au^\nu\|}{\|b\|}. \quad (5.91)$$

The error estimate (5.91) will be used to determine whether it is time to stop the multigrid iteration for solving the discretized equations of the linear diffusion (3.45), which results from an operator splitting procedure for the singularly perturbed reaction-diffusion equations, such as the bidomain model (3.1) or the monodomain model (3.2).

On the other hand, it is also well-known that the iteration

$$u^{\nu+1} = u^\nu + B^{-1}(b - Au^\nu)$$

for the linear system $Au = b$ has the following absolute error estimate

$$\|u - u^{\nu+1}\| \leq \frac{\rho}{1 - \rho} \|u^{\nu+1} - u^\nu\| \quad (5.92)$$

if the energy norm $\rho \equiv \|I - B^{-1}A\|_a$ of the error propagation operator $E = I - B^{-1}A$ is less than one, i.e.,

$$\rho = \|I - B^{-1}A\|_a < 1.$$ 

Here, the energy norm $\| \cdot \|_a$ is induced by the inner product $(A \cdot, \cdot) = a(\cdot, \cdot)$. 

162
For the standard $V$-cycle multigrid algorithm, we know that the energy norm of the error propagation operator $E = I - B^{-1}A$ is bounded by a less-than-one positive number, i.e.,

$$\|E\|_a = \|I - B^{-1}A\|_a \leq 1 - \frac{C}{J} \quad (5.93)$$

with

$$C = \frac{1}{(1 + c_1 + c_2 C_s^{1/2})^2}$$

defined in (5.61) (or see the original paper by Bramble [43]). Substituting (5.93) into the absolute error estimate (5.92) yields

$$\|u - u^{\nu+1}\|_a \lesssim J \|u^{\nu+1} - u\|_a. \quad (5.94)$$

In addition, by an inverse estimate again, we have

$$\|u^{\nu+1} - u\|_a \lesssim \frac{1}{h} \|u^{\nu+1} - u\|. \quad (5.95)$$

Combining (5.95) and (5.94) yields the following absolute error estimate

$$\|u - u^{\nu+1}\|_a \lesssim \frac{J}{h} \|u^{\nu+1} - u\|. \quad (5.96)$$

Moreover, if a typical magnitude (scale) $S$ of the solution $u$ is provided for the iteration, say,

$$S = |u|_\infty, \quad (5.97)$$

then, by the norm equivalence in a finite dimensional space and the first inequality in (5.89), we obtain

$$|\Omega| S^2 < (1 + \epsilon \tau) |\Omega| S^2 \lesssim (1 + \epsilon \tau) (v, v) \lesssim (Av, v) = \|v\|_a^2$$

i.e.,

$$|\Omega|^{1/2} S \lesssim \|v\|_a, \quad (5.98)$$

163
for any $v$ in the finite element space $V$. Here, $|\Omega|$ denotes the area (2D) or volume (3D) of the domain $\Omega$. Finally, combining the inequality (5.98) with the absolute error estimate (5.96) gives us another relative error estimate

$$\frac{\|u - u^{\nu+1}\|_a}{\|u\|_a} \lesssim \frac{J}{|\Omega|^{1/2} S h} \|u^{\nu+1} - u^{\nu}\|.$$

(5.99)

It is remarkable that the constant $h$, which appears in each of the error estimates above, can be replaced by the mesh parameter $h$ of the main grid $G_J$ on the finest level since the main grid $G_j$ on each level $\ell_j$ in our AMR algorithm is shape-regular [34, 57], which implies that $h \approx h$.

In summary, we can use either the relative error estimates (5.91), (5.99) or the absolute error estimate (5.96) to determine whether it is time to terminate the $V$-cycle multigrid iteration for solving the discretized equations of the linear diffusion (3.45), which results from an operator splitting procedure for the singularly perturbed reaction-diffusion equations, such as the bidomain model (3.1) or the monodomain model (3.2) (see Figure 5.6). In the implementation, we use the relative error estimate (5.91).
Error estimates for the V-cycle multigrid iteration:

1. Relative error estimate I:
\[
\frac{\| u - u^\nu \|}{\| u \|} \lesssim \max \left\{ 1, \frac{\epsilon \tau}{h^2} \right\} \frac{\| b - Au^\nu \|}{\| b \|},
\]

2. Absolute error estimate:
\[
\| u - u^{\nu+1} \|_a \lesssim \frac{J}{h} \| u^{\nu+1} - u^{\nu} \|.
\]

3. Relative error estimate II:
\[
\frac{\| u - u^{\nu+1} \|_a}{\| u \|_a} \lesssim \frac{J}{|\Omega|^{1/2} S h} \| u^{\nu+1} - u^{\nu} \|.
\]

Figure 5.6: Error estimates for the V-cycle multigrid iteration
Chapter 6

Numerical Experiments

Numerical results with the AMR algorithm are presented in this chapter. First, section 6.1 describes the tolerances, parameters and modules used in the experiments. It is demonstrated in section 6.2 that the convergence rate of the algorithm is second-order by a few singularly perturbed reaction-diffusion problems, whose exact solutions are known. Next, section 6.3 shows numerical accuracy of the algorithm by a 1D FitzHugh-Nagumo model problem with an appropriate choice of the tolerance for tagging of cells. Computational times spent in the numerical experiments for the FitzHugh-Nagumo model problem in both 1D and 2D are presented in section 6.4. The results indicate that the AMR algorithm has a very good scaling property and hence demonstrate the efficiency of the algorithm. Section 6.5 illustrates the geometry flexibility of the algorithm. In the last section, a few snapshots from the experiments with the Luo-Rudy phase one bidomain model are presented, which shows the promising application of the AMR algorithm to realistic simulations.

6.1 Introduction

The tolerances, parameters and modules of the AMR algorithm proposed in the thesis work are described in this section. By default, all of the experiments presented in this chapter use the same tolerances, parameters and modules unless explicitly stated.

In our experiments, the second coarsest level is always generated by uniform refinement of the coarsest level grid. All other fine levels are locally, dynamically and automatically generated by the algorithm. The refinement ratio is set as \( r = 2 \) and the regrid interval is chosen to be \( \kappa = 2 \).
By the algorithm, the second-order Strang splitting is employed to decouple the nonlinear reactions from the linear diffusion. In each fine time step during the adaptive integration process, the nonlinear reaction(s) on the current fine level is(are) first adaptively integrated by a half time step with the second order SDIRK scheme. Then, the linear diffusion is advanced by a full fine time step with the Crank-Nicolson (conforming) finite element method. The initial values for the linear diffusion on the current finest level are given by those updated by the first half reaction integration while those on coarse levels are simply determined using time interpolation to achieve fully space-time adaptive integration [181] (for numerical results using updated data by the first half reaction integration as initial values for the linear diffusions, see Appendix A). The resulting linear systems on composite grids are solved by the standard $V$-cycle multigrid iteration even though in one space dimension they are just tri-diagonal, simple enough to use a direct solver. By the Strang splitting, after integration of the linear diffusion with a full fine time step, the nonlinear reaction(s) is(are) adaptively integrated by another half time step with the second-order SDIRK scheme.

The tolerance for the conjugate gradient iteration on the coarsest level is $10^{-8}$ (all of our simulations use double-precision). The iteration stops when the ratio of the discrete $\ell^2$ norm of the residual to that of the right hand side is less than $10^{-8}$. By a discrete $\ell^2$ norm, we mean

$$\|v\|_{\ell^2} = \sqrt{\sum_{i=1}^{n} v_i^2}$$

for a vector $v \in \mathbb{R}^n$. In the conjugate gradient solver, if the discrete norm of the right hand side is already less than $10^{-8}$, the solver returns immediately with a solution equal to zero. In the current implementation of the AMR algorithm, the conjugate gradient iteration is used only for systems on the coarsest level.
The multigrid iteration tolerance is computed by an empirical formula, proposed by Trangenstein [181]:

$$tol_{\text{multigrid}} = \min \left\{ 10^{-3}, \max \left\{ \sqrt{\epsilon_{\text{mach}}}, \Delta x^2 \right\} \right\},$$

where $\epsilon_{\text{mach}}$ is the machine roundoff. In effect, the coefficient of the quadratic term in this formula is chosen to be much less than that in an a priori error estimate for our algorithm, such as the coefficient $C$ in (3.58). In the experiments, such a coefficient $C$ can be estimated. The multigrid iteration continues unless the product of the estimated matrix condition number and the ratio of the computed discrete residual norm to the discrete norm of the right hand side is greater than the tolerance; see (5.91). The matrix condition number is estimated by

$$\kappa(A) \sim \max \left\{ 1, \frac{\epsilon \Delta t}{\Delta x^2} \right\},$$

where $A$ is the matrix resulting from the discretization of linear diffusion and $\epsilon$ is a typical size scale of the diffusion tensor. Due to the level-dependent iteration tolerance, it is reasonable that the number of multigrid iterations depends on the number of levels involved. In the relaxation (smoothing) step of the multigrid algorithm, the Gauss-Seidel iteration is used for data on nodes interior to the main grid of each level while the Jacobi iteration is used for data on nodes along the coarse-fine interface boundaries. The relaxation, consisting of one step of such a mixed iteration, occurs only on the current and its coarser level, at most two levels, by the theory proposed by Bramble [41] and what we described in section 5.5 (see Figure 5.4).

The tolerance used in the adaptive integration of reactions with the second SDIRK scheme is similarly computed by another empirical formula:

$$tol_{\text{SDIRK}} = \min \left\{ 10^{-3}, \max \left\{ \sqrt{\epsilon_{\text{mach}}}, 5 \Delta x^2 \right\} \right\},$$
with $\epsilon_{\text{mach}}$ the machine roundoff. The Richardson extrapolation stops when the relative error in (3.39) is greater than the tolerance. The tolerance in the nonlinear Newton solver for the SDIRK scheme is fixed to be $10^{-8}$ (it will be adjusted to be dependent of the mesh parameter of the current level grid in the near future for better performance). The nonlinear solver stops when the absolute value of the determinant of the Jacobian matrix (derivative for the 1D test problem) is less than $10^{-8}$ or the relative correction made is less than the tolerance.

Notice that the overall numerical error is bounded by or on the order of the global relative error tolerance used in the Richardson extrapolation based error estimation procedure for tagging of coarse cells. To study the convergence rate of the AMR algorithm in section 1, the global relative error tolerance is chosen to be small and level-dependent. It is computed by the following formula

$$tol_{\text{AMR}} = \min \left\{ 10^{-1}, \max \left\{ 10^{-4}, \Delta x^2 \right\} \right\}.$$  \hspace{1cm} (6.2)

In section 2 and section 3, the tolerance will be changed to be a fixed larger number to study the efficiency of the algorithm while the numerical results are accurate enough for the purpose of practical applications. Cells are tagged if the relative error in (4.3) is greater than the tolerance $tol_{\text{AMR}}$. In our current implementation, the data $w^c(\nu \Delta t)$ on coarse levels used to perform error estimation by (4.3) are obtained by linear interpolation. Strictly, the interpolation used should have accuracy at least one order higher than the overall algorithm. The linear interpolation may influence (enlarge) the size of the refined region even though our numerical experiments demonstrate that it is not a big problem (see Appendix A and Appendix B). In the future, a higher order interpolation or a procedure similar to Berger-Oliger’s strategy [24, 27] shall be incorporated.

All of the numerical experiments in the thesis work are performed in a Linux machine with an AMD Athlon XP 2600+ 2.08 GHz CPU and 1.5 GB memory.
6.2 Convergence Rate

In our AMR algorithm for the reaction-diffusion problems, numerical errors mainly come from five parts: (a) the operator splitting; (b) the time integration of the nonlinear reactions; (c) the discretization of the linear diffusion; (d) the nonlinear Newton solver; and (e) the multigrid iteration for linear systems. The errors from parts (b) and (d), related to integration of nonlinear reactions on coarse levels, are controlled by the adaptive integration process with the SDIRK scheme. The operator splitting error in part (a) on coarse levels is controlled by the local mesh refinement with the Richardson extrapolation based error estimation procedure. The multigrid iteration error in part (e) can also be controlled by the iteration tolerance $tol_{\text{multigrid}}$.

With the tolerances described in the previous section, the errors from these five parts, except (b) on coarse levels, are all controlled to be on the order of that resulting from the finite element discretization of the linear diffusion. The overall (global) error is principally contributed by the splitting error on the finest levels and the finite element discretization error on all levels, if nonlinear reactions on the finest levels are integrated to be arbitrarily high order accurate or the error is controlled by the adaptive SDIRK scheme too. In effect, on the finest levels, there is no need to use adaptive time integration for nonlinear reactions with the SDIRK scheme since it gives second-order accurate results and we have no intent to control errors there. This will definitely enhance the algorithm performance.

In the next, we would like to demonstrate by numerical experiments that the convergence rate of the AMR algorithm proposed is second-order for the reaction-diffusion problems.
Table 6.1: AMR for the 1D test problem at time $t = 0.25$

<table>
<thead>
<tr>
<th>#refinement levels</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>#finest cells</td>
<td>26</td>
<td>44</td>
<td>88</td>
<td>174</td>
<td>348</td>
</tr>
<tr>
<td>#multigrid iterations</td>
<td>2</td>
<td>3</td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>error in $L^2$ norm</td>
<td>0.0065727</td>
<td>0.0014705</td>
<td>0.0003716</td>
<td>0.0000941</td>
<td>0.0000249</td>
</tr>
</tbody>
</table>

Table 6.2: AMR for the 1D test problem at time $t = 0.5$

<table>
<thead>
<tr>
<th>#refinement levels</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>#finest cells</td>
<td>26</td>
<td>44</td>
<td>88</td>
<td>174</td>
<td>348</td>
</tr>
<tr>
<td>#multigrid iterations</td>
<td>2</td>
<td>3</td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>error in $L^2$ norm</td>
<td>0.0141850</td>
<td>0.0031800</td>
<td>0.0008200</td>
<td>0.0002161</td>
<td>0.0000643</td>
</tr>
</tbody>
</table>

Table 6.3: AMR for the 1D test problem at time $t = 1$

<table>
<thead>
<tr>
<th>#refinement levels</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>#finest cells</td>
<td>26</td>
<td>46</td>
<td>88</td>
<td>174</td>
<td>348</td>
</tr>
<tr>
<td>#multigrid iterations</td>
<td>2</td>
<td>3</td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>error in $L^2$ norm</td>
<td>0.0291523</td>
<td>0.0065811</td>
<td>0.0017219</td>
<td>0.0004663</td>
<td>0.0001512</td>
</tr>
</tbody>
</table>

6.2.1 The Case in One Space Dimension

Example 6.1: Let us consider the following scalar singularly perturbed reaction-diffusion equation

$$\frac{\partial u}{\partial t} = \epsilon \Delta u + \frac{2}{\epsilon} (1 - u)u^2$$

for $t > 0$, $x \in (0, 4)$,

with a small positive parameter $\epsilon$. In the numerical experiments, we take $\epsilon = 1/32$. It is known that this equation together with appropriate initial and boundary conditions has traveling wave solutions. The initial values are given by

$$u(t, x) = \frac{1}{1 + e^{(x-t-1.5)/\epsilon}},$$

171
Figure 6.1: Adaptively refined grids for the 1D test problem at time \( t = 1 \) which actually provides an exact solution to the equation. Noticing that the exact solution are almost constant close to the endpoints of the computational domain, we can apply no-flux (homogeneous Neumann) boundary conditions there. Hence, the convergence rate of our AMR algorithm for the bidomain/monodomain models with homogeneous Neumann boundary conditions can be demonstrated.

The problem is solved on the interval \([0, 4]\) in space from 0 to 1 in time with the AMR algorithm. In the experiments, the coarsest level grid has 16 cells of equal size with \( \Delta x = 0.25 \). The time step size is chosen to be equal to the mesh parameter, i.e., \( \Delta t = \Delta x \). The choice of time step size guarantees that the wave front never travels outside the region covered by fine levels since the traveling wave presented by the solution has speed one.

Figure 6.1 shows the adaptively refined grids for the 1D test problem at time \( t = 1 \). Numerical results are presented in Tables 6.1-6.3. The first row in each table denotes the maximum number of refinement levels. The second row is the finest grid size. The third row contains the number of cells on the finest level. The next row has
the number of the $V$-cycle multigrid iterations on the composite grids. The last row represents the numerical error in $L^2$ norm. The results indicate that the convergence rate of the AMR algorithm is second-order for the 1D test problem.

In fact, we also compute the locations of the traveling wave front; see Table 6.4. We look at the numerical results at time $t = 1$ and linearly interpolate the solution versus space coordinates to determine the spatial location of $u = 0.5$. It seems that the numerical solution wave travels slower than the exact one, approaching the exact front location ($x = 2.5$) from the left for the test problem. The results further indicate that the convergence rate of our AMR algorithm is second-order.

It is noticeable from the numerical results that fine level grids are almost uniformly refined. This is mainly due to the small tolerance $tol_{AMR}$ defined in (6.2). To study the convergence rate as more levels of refinement are added, this tolerance must be small enough such that the errors from coarse level are well controlled. As a supplement, we present numerical results with three times larger tolerance for tagging of cells in Appendix C. In those experiments, we use exact errors to tag cells to eliminate the effect of the linear interpolation used in the error estimation procedure (see Appendix B). The phenomenon of almost uniformly (“overly”) refined grids on fine levels is still observed there.

<table>
<thead>
<tr>
<th>#refinement levels</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>#finest cells</td>
<td>26</td>
<td>46</td>
<td>88</td>
<td>174</td>
<td>348</td>
</tr>
<tr>
<td>front location</td>
<td>2.487523</td>
<td>2.497253</td>
<td>2.499321</td>
<td>2.499844</td>
<td>2.499975</td>
</tr>
<tr>
<td>location deviation</td>
<td>0.012477</td>
<td>0.002747</td>
<td>0.000679</td>
<td>0.000156</td>
<td>0.000025</td>
</tr>
</tbody>
</table>

Table 6.4: Location of the 1D traveling wave front at $t = 1$
6.2.2 The Cases in Two Space Dimensions

To demonstrate the convergence rate of the AMR algorithm for the reaction-diffusion problems with homogeneous Neumann boundary conditions in higher space dimensions, we still consider scalar equations with exact solutions known. In this subsection, two examples demonstrating the second-order convergence rate of the AMR algorithm are presented. One involves a planar traveling wave and another one has a circular wave solution.

**Example 6.2:** Let us again consider the scalar reaction-diffusion equation

$$\frac{\partial u}{\partial t} = \epsilon \Delta u + \frac{2}{\epsilon}(1 - u)u^2 \quad \text{for } t > 0, \ x \in (0, 4)^2.$$  \hspace{1cm} (6.3)

Here, the parameter $\epsilon$ has the same value as in the previous 1D test case. The initial values in the 2D experiments are given by

$$u(t, x) \equiv u(t, x_1, x_2) = \frac{1}{1 + e^{(x_1 - t - 1.5)/\epsilon}},$$

which is very approximately an exact solution to the equation (6.3) with homogeneous Neumann boundary conditions when time is not too large (e.g., $t < 1$). The solution represents a planar traveling wave of speed one.

The problem is solved on the square rectangular domain $[0, 4] \times [0, 4]$ from 0 to 1 in time. The coarsest level grid has $16 \times 16 = 256$ cells of equal size with $\Delta x = 0.25$. The time step size is always equal to the mesh parameter, i.e., $\Delta t = \Delta x$. Here, the mesh parameter $\Delta x$ is defined to be the side length of an element. The adaptively refined grids and solution iso-contours from the simulation with six levels for the 2D test problem at time $t = 1$ is shown in Figure 6.2. The solution contours range from 0.0001 to 0.9999 with forty equally spaced values. Numerical results are presented in Tables 6.5-6.7.
Figure 6.2: Adaptively refined grids and solution iso-contours (0.0001 – 0.9999) for the first 2D test problem at time $t = 1$
Table 6.5: AMR for the first 2D test problem at time \( t = 0.25 \)

<table>
<thead>
<tr>
<th>#refinement levels</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>finest cell size</td>
<td>4/64</td>
<td>4/128</td>
<td>4/256</td>
<td>4/512</td>
<td>4/1024</td>
</tr>
<tr>
<td>#finest cells</td>
<td>1024</td>
<td>3072</td>
<td>10752</td>
<td>40960</td>
<td>167936</td>
</tr>
<tr>
<td>#multigrid iterations</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>error in ( L^2 ) norm</td>
<td>0.0401401</td>
<td>0.0128399</td>
<td>0.0029821</td>
<td>0.0007559</td>
<td>0.0001966</td>
</tr>
</tbody>
</table>

Table 6.6: AMR for the first 2D test problem at time \( t = 0.5 \)

<table>
<thead>
<tr>
<th>#refinement levels</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>finest cell size</td>
<td>4/64</td>
<td>4/128</td>
<td>4/256</td>
<td>4/512</td>
<td>4/1024</td>
</tr>
<tr>
<td>#finest cells</td>
<td>1024</td>
<td>3072</td>
<td>10752</td>
<td>40960</td>
<td>167936</td>
</tr>
<tr>
<td>#multigrid iterations</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>error in ( L^2 ) norm</td>
<td>0.0761228</td>
<td>0.0275626</td>
<td>0.0064903</td>
<td>0.0017197</td>
<td>0.0004787</td>
</tr>
</tbody>
</table>

Table 6.7: AMR for the first 2D test problem at time \( t = 1 \)

<table>
<thead>
<tr>
<th>#refinement levels</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>finest cell size</td>
<td>4/64</td>
<td>4/128</td>
<td>4/256</td>
<td>4/512</td>
<td>4/1024</td>
</tr>
<tr>
<td>#finest cells</td>
<td>1024</td>
<td>3072</td>
<td>10752</td>
<td>40960</td>
<td>167936</td>
</tr>
<tr>
<td>#multigrid iterations</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>error in ( L^2 ) norm</td>
<td>0.1781071</td>
<td>0.0565132</td>
<td>0.0134919</td>
<td>0.0036804</td>
<td>0.0010847</td>
</tr>
</tbody>
</table>

In the experiments, all parameters, tolerances and modules are the same as those used in the 1D case except that the tolerance for the multigrid iteration is empirically adjusted to be tenth of that for the 1D test case. Specifically, the tolerance for the multigrid iteration is given by

\[
tol_{\text{multigrid}} = \min \{10^{-3}, \max \{\sqrt{\epsilon_{\text{mach}}}, 0.1 \Delta x^2\}\}. \tag{6.4}
\]

The numerical results demonstrate that the convergence rate of the AMR algorithm is second-order.

**Example 6.3:** Let us consider the following nonlinear reaction-diffusion equation:

\[
\frac{\partial u}{\partial t} = \epsilon \Delta u + \frac{1}{\epsilon} u(1 - u) \left\{ 2u + \frac{\epsilon}{|x|} \right\} \quad \text{for} \ t > 0, \ x \in (0,4)^2. \tag{6.5}
\]
The initial values are given by

\[ u = \frac{1}{1 + e^{(|x| - t - 1) / \epsilon}}, \]

which is also very approximately an exact solution of the problem with homogeneous Neumann boundary conditions when time is not too large (e.g., \( t < 2 \)). The solution represents a circular traveling wave.

The problem is solved on the spatial domain \((0, 4)^2\) from 0 to 2 in time. The coarsest grid has \(16 \times 16 = 256\) cells of equal size with \(\Delta x = 0.25\). The time step size is equal to the mesh parameter, \(\Delta t = \Delta x\) on each level. All parameters, tolerances and modules used in the experiments are the same as those for Example 6.2.

Numerical results are summarized in Tables 6.8-6.10. A plot of the adaptively refined grids and solution contours at time \(t = 2\) with five levels are shown in Figure 6.3. The values of the solution contours range from 0.0001 to 0.9999 with forty equally spaced values.

From the results, we may also conclude that the AMR algorithm in the 2D case yields second-order accurate results as well as in the 1D case.

Similarly, it is also noticeable from the numerical results that fine level grids are almost uniformly refined. This is because, in order to observe the second-order convergence on fine levels, the tolerance \(\text{tol}_{\text{AMR}}\) (6.2) must be small enough such that the errors from coarse levels are well controlled. As a supplement, we present numerical results with three times larger tolerance for tagging of cells in Appendix C. In those experiments, we only use exact errors to tag cells to eliminate the effect of the linear interpolation used in the error estimation procedure (see Appendix B). The phenomenon of almost uniformly (“overly”) refined grids on fine levels is still observed.
Figure 6.3: Adaptively refined grids and solution iso-contours (0.0001 − 0.9999) for the second 2D test problem at time $t = 2$
6.3 Numerical Accuracy

In the experiments presented below, all other parameters, tolerances and modules are the same as those described in section 6.1 except that the tolerance used in the Richardson extrapolation based error estimation procedure for tagging of cells is chosen to be much larger than that in the previous section. Following Trangenstein [181], we choose the relative error tolerance to be $tol_{AMR} = 0.1$. It is remarkable that the choice of $tol_{AMR}$ determines the numerical accuracy and efficiency of the AMR algorithm. The smaller the tolerance, more accurate but less efficient is the AMR algorithm. Vice versa, the larger the tolerance, more efficient but less accurate is the AMR algorithm. In practice, the “optimal” value of the tolerance $tol_{AMR}$ to achieve the best accuracy and efficiency depends on the problem to be solved.

**Example 6.4:** The following version of the dimensionless FitzHugh-Nagumo monodomain model is considered in the numerical experiments:

\[
\frac{\partial v}{\partial t} = \varepsilon \nabla \cdot \nabla v + \lambda (q - v(1 - v)(v - \theta)),
\]

\[
\frac{\partial q}{\partial t} = \alpha v - \beta q,
\]

for $t > 0$ and $x \in (0, 1)$. Here, $\varepsilon > 0$ is the diffusion parameter and $\lambda < 0$ is the reaction (eigenvalue) parameter. It is known that this model has traveling wave solutions [105, 182] with an appropriate choice of parameters: $\varepsilon$, $\lambda$, $\theta$, $\alpha$ and $\beta$. In our computations, the parameters are chosen such that the traveling waves involved have positive speed but less than one. Specifically, the diffusion constant $\varepsilon = 0.01$, the eigenvalue parameter $\lambda = -100$. Other parameters are given by $\theta = 0.25$, $\alpha = 0.16875$ and $\beta = 1.0$. The initial values are given by the following smooth data

\[v = \frac{1}{1 + e^{-50(x-0.2)}} \quad \text{and} \quad q = 0,\]
and homogeneous Neumann boundary conditions are applied. The time step size is chosen to be equal to the grid size on each level, i.e., \( \Delta t = \Delta x \). This selection of time step size guarantees that the reaction fronts never travel more than one cell per time step since the wave speed is always less than one. The coarsest level grid has eight elements of equal size with \( \Delta x = 1/8 \). The problem is solved on the spatial interval \((0, 1)\) from 0 to 2 in time.

For the purpose of comparison, we first run a simulation on uniformly refined grids with ten levels of refinement. The grid size on the finest uniform grid is \( \Delta x = 1/4096 \). By the demonstrated convergence rate of the algorithm, we may think of the solution from this experiment as an “exact” solution. The membrane potential of the model problem at time \( t = 2 \) is shown in Figure 6.4. The background grid ranges from 0 to 1 in the horizontal direction and from \(-0.2\) to 1 in the vertical direction.

Next we performed numerical simulations with four to nine levels of adaptive refinement, respectively. Corresponding membrane potentials at time \( t = 2 \) are shown in Figure 6.5.

Comparing the AMR and uniform refinement results, we may conclude that they agree very well and become better as more levels of mesh refinement are added in the simulation. Here, we also observe that the traveling waves approach to the exact one from the left as more levels of refinement are added.

The computational times for the 1D experiments will be reported and discussed in the next section together with those in higher space dimensions.
6.4 Algorithm Efficiency

The 1D experiments presented in section 6.3 demonstrate that the choice of the tolerance for tagging of cells ($\text{tol}_{\text{AMR}} = 0.1$) should give us accurate enough results.

In this section, we keep using the same tolerance $\text{tol}_{\text{AMR}}$ as in section 6.3. In the one space dimensional case, the tolerance for multigrid iteration is given by (6.1). In the two space dimensional case, the tolerance for multigrid iteration is smaller, given by (6.4). All other parameters, tolerances and modules in the AMR algorithm are the same as those described in section 6.1.

6.4.1 The Case in One Space Dimension

Example 6.4 (continued): The computational times in the experiments with $\text{tol}_{\text{AMR}} = 0.1$ for the 1D FitzHugh-Nagumo model problem are shown in Tables 6.12 and 6.13, corresponding to adaptive and uniform mesh refinement respectively. The first row in each table denotes the maximum number of refinement levels. The

\[ \Delta x = \Delta t = 1/4096 \]
Figure 6.5: Membrane potentials of the 1D FitzHugh-Nagumo model at time $t = 2$ (with the space-time adaptive integration algorithm and $tol_{AMR} = 0.1$)
second row has the finest grid size. The third row contains the number of finest level cells at time $t = 2$. The next three rows have the computational times respectively spent (a). in the adaptive integration of nonlinear reactions with the second order SDIRK scheme, (b). in the V-cycle multigrid iteration for linear diffusion, and (c). in the adaptive mesh refinement including tagging of coarse cells and regridding of fine levels, respectively. The last row in the table corresponding to uniform refinement and the second-last row in the tables corresponding to adaptive refinement represent the total computational times.

Comparing the results with eight levels of refinement, we find that the AMR simulation gains 4.9 times speed-up. It seems that the AMR simulation with maximum nine levels of refinement even gains much more speed-up. The computational speed-up is about 12.7. Mostly this is because of the relatively large tolerance $tol_{AMR} = 0.1$, which tags only a few cells on the second finest levels. However, we should keep in mind that the adaptive results may not be as accurate as the uniform results. For more discussion on accuracy with larger tolerances and more levels of refinement, see Appendix C.

For the purpose of comparison, we also performed numerical experiments with smaller tolerance for tagging of cells, i.e., $tol_{AMR} = 0.02$ and $tol_{AMR} = 0.05$, respectively. The computational times are shown in Tables 6.14 and 6.15. It is obvious that smaller tolerance implies more cells to be refined on each level and hence more computational times. The experiment with $tol_{AMR} = 0.05$ and nine levels of refinement only achieves about 5.3 times speed-up. The gain in efficiency in the experiment with $tol_{AMR} = 0.02$ and nine levels of refinement is even smaller with merely 3.6 times speed-up.

From the results shown in Tables 6.14-6.12, we observe that most of the computational times are spent in the integration of nonlinear reactions. Only a small fraction
is spent in the multigrid solver for the linear diffusion. The results in Table 6.12 corresponding to \( tol_{AMR} = 0.1 \) show that, in the experiment with maximum eight levels of adaptive refinement, the integration of nonlinear reactions uses about 87.7\% of the computational time and the multigrid iteration for linear diffusion uses only about 7.7\%.

### 6.4.2 The Case in Two Space Dimensions

**Example 6.5:** We still consider the dimensionless FitzHugh-Nagumo monodomain model

\[
\frac{\partial v}{\partial t} = \varepsilon \nabla \cdot \nabla v + \lambda (q - v(1 - v)(v - \theta)), \tag{6.7a}
\]

\[
\frac{\partial q}{\partial t} = \alpha v - \beta q, \tag{6.7b}
\]

for \( t > 0 \) and \( \mathbf{x} = (x_1, x_2)^T \in (0, 1)^2 \), with the parameters: \( \varepsilon = 0.01 \), \( \lambda = -100 \), \( \theta = 0.25 \), \( \alpha = 0.16875 \) and \( \beta = 1.0 \). The initial values are given by the smooth data

\[
v = \frac{1}{1 + e^{-50(\sqrt{x_1^2 + x_2^2} - 0.1)}} \quad \text{and} \quad q = 0,
\]

and homogeneous Neumann boundary conditions are applied. The problem is solved on the unit rectangular square \((0, 1)^2\) from 0 to 5 in time with both adaptive and uniform mesh refinement.

In the experiments, the coarsest level grid has \( 8 \times 8 = 64 \) elements of equal size with \( \Delta x = 1/8 \). The time step size is chosen to be equal to the grid size on each level, i.e., \( \Delta t = \Delta x \), to guarantee that the reaction front never travels more than one cell per time step since the wave speed is less than one.

The adaptively refined grids and iso-contours of the membrane potential are shown in Figure 6.6. The contours range from 0.01 to 0.99 with 10 equally distributed values.
The computational times are shown in Tables 6.16 and 6.17. The speed-up of the adaptive integration, with six levels of refinement \((\Delta x = 1/256)\), over corresponding uniform computation on a \(256 \times 256\) grid, is about 3.46. It is observed that most of the computational times are spent in the adaptive integration of nonlinear reactions with the SDIRK. Note that the tolerance in the Newton solver for the SDIRK scheme is fixed to be \(10^{-8}\) in our current implementation. If we instead use a level-dependent or mesh-dependent tolerance, the percentage of the time used in reaction integration will be smaller. The time spent in the multigrid iteration is only about 13.7\% in the adaptive experiment with six levels of refinement. In fact, in the experiments, the \(V\)-cycle multigrid algorithm for linear systems perform very well, typically using 2–4 iterations to converge to the tolerance \(tol_{\text{multigrid}}\) described in (6.4) even though there is no preconditioned conjugate gradient iteration used on fine levels.

The data in Table 6.16 also show that the time used in tagging of coarse cells and regridding of fine levels only occupies a small fraction. For example, in the experiment with six levels of refinement, the percentage is just about 1.2\%. This is because we use the very cheap error estimation procedure by linear interpolation to obtain coarse level data assuming the data are piece-wise linear in time. If we use a higher order interpolation or Berger-Oliger’s strategy, the adaptive mesh refinement will be more expensive.

In summary, with appropriate choice of the tolerance \(tol_{\text{AMR}}\), we demonstrated that our AMR algorithm has a good scaling property in both one and two dimensional spaces. There is no wonder that, in simulations with realistic membrane models which have much sharper wave fronts, corresponding to the depolarization phase, the speed-up gained by the AMR algorithm will be much more, especially when the tissue is in its plateau and resting phases (see Figures 1.1, 1.2, and 1.3). On the other hand, there is also no wonder that the AMR algorithm will lose its efficiency and advantages
in the case that the cardiac tissue is in a state of chaotic electrical activity or when multiple spiral or scroll waves are present in the space; see Appendices D and E.

6.4.3 The Case in Three Space Dimensions

The AMR algorithm proposed in the thesis work is naturally extended to the case in three space dimensions.

**Example 6.6:** In three space dimensions, we keep considering the dimensionless FitzHugh-Nagumo monodomain model

\[
\begin{align*}
\frac{\partial v}{\partial t} &= \epsilon \nabla \cdot \nabla v + \lambda (q - v(1 - v)(v - \theta)), \\
\frac{\partial q}{\partial t} &= \alpha v - \beta q,
\end{align*}
\]  \hspace{1cm} (6.8a) \hspace{1cm} (6.8b)

for \( t > 0 \) and \( \mathbf{x} = (x_1, x_2, x_3)^T \in (0,1)^3 \), with the parameters: \( \epsilon = 0.01, \lambda = -100, \theta = 0.25, \alpha = 0.16875 \) and \( \beta = 1.0 \). The initial values are given by

\[
v = \begin{cases} 
1 & \text{if } \sqrt{(x_1 - 1)^2 + (x_2 - 1)^2 + (x_3 - 1)^2} < 0.1 \\
0 & \text{otherwise}
\end{cases} \quad \text{and} \quad q = 0
\]

and homogeneous Neumann boundary conditions are applied. The problem is solved on the unit cube \((0,1)^3\) with our AMR algorithm. Unfortunately, due to the limited computer memory, we were not able to perform simulations with many levels of refinement or run the program for a long time as the wave fronts occupy most of the computational domain. Up to now, no accuracy and efficiency studies have been made yet. To demonstrate that the AMR algorithm works in three space dimensions as well as in the lower dimensional cases, we presents a few snapshots in Figure 6.7 from the simulation performed with six levels of refinement and the coarsest level grid having \( 4 \times 4 \times 4 = 64 \) cells of equal size. The simulation amounts the uniform computation on a \( 128 \times 128 \times 128 \) grid. All of other parameters, tolerances and modules used are the same as those for Example 6.5.
Figure 6.6: Adaptively refined grids and iso-contours (0.01 − 0.99) of the membrane potentials for the 2D FitzHugh-Nagumo model.
For scroll wave results from the simulation of the 3D FitzHugh-Nagumo model with our space-time adaptive algorithm, see Appendix E.

6.5 Geometry Flexibility

The AMR algorithm works for problems on complex domains as well as on logically rectangular grids due to the flexible grid representation (see section 4.3.1). In this section, we present results from a simulation on a complex 2D domain.

Example 6.5 (continued): The dimensionless FitzHugh-Nagumo monodomain model problem (6.8) is also solved on the unit square with four equally distributed holes of radius 0.1. See Figure 6.8 for a few snapshots from the experiment. In this simulation, a second stimulus given by

\[
I_{\text{stim}} = \begin{cases} 
1 & \text{if } \sqrt{(x_1 - 0.5)^2 + (x_2 - 0.5)^2} < 0.2 \\ 
0 & \text{otherwise}
\end{cases}
\]

is applied to the membrane potential around the center of the domain at time \( t = 4 \).

6.6 Applications

It is straightforward to apply our AMR algorithm to more realistic simulations of electrical wave propagation in the heart. We have started and will continue working on the applications.

The simulations with the Hodgkin-Huxley, Beeler-Reuter, Fenton-Karma, Saeffer Phase II, Luo-Rudy phase I and Luo-Rudy dynamic models are all almost completely implemented except that the implicit integration of nonlinear reactions works only with the first-order Backward Euler method without adaptivity. The adaptive integration with the backward Euler, the second-order and third-order SDIRK methods still needs to be debugged and validated (e.g., negative Calcium concentration is computed in 2D computations).
(a) at time $t = 0.96$

(b) at time $t = 1.01$
Figure 6.7: Adaptively refined grids and potential iso-surfaces \( (v = 0.5) \) for the 3D FitzHugh-Nagumo problem
(a) at time $t = 0$

(b) at time $t = 0.956711$

(c) at time $t = 1.95647$

(d) at time $t = 2.95624$

(e) at time $t = 3.956$

(f) at time $t = 4.95576$
Figure 6.8: The FitzHugh-Nagumo monodomain model on complex domain (2D)
In one space dimension, the Luo-Rudy phase I model on a 2 cm-length tissue is simulated for a beat (BCL = 300 ms) with both the adaptive and uniform refinement methods. In the AMR simulation, the coarsest grid has 16 cells of equal size ($\Delta x = 1/8$) and totally six levels of refinement are involved. Due to the lower-order (first-order backward Euler) integration of the nonlinear reactions in the current implementation, the expected global order $p$ of the scheme is changed to be $p = 1$ in the error estimation (4.3) for tagging of cells. All other parameters, tolerances and modules are the same as those for the 1D FitzHugh-Nagumo model in Example 6.4. In the experiments, a current stimulus is applied at the left end of the tissue and the action potential are collected at a position, 0.5 cm away from the stimulation point; see Figure 6.9. It seems that the adaptive and uniform results agree very well, almost identical to each other except that there is slight difference in the upstroke phase but the difference is less than 1%. On the other hand, it is observed that, the adaptive simulation uses only 42.95 seconds of computer time while the corresponding uniform refinement simulation on a grid with 512 cells uses 1460.26 seconds. The speed-up gained by the AMR algorithm is about 34.0.

We also performed numerical experiments for the Beeler-Reuter model on the 2 cm-length tissue over a period of one beat (BCL = 400 ms). All other parameters, modules and environmental settings are the same as those in the experiment for the Luo-Rudy phase I model except that the action potential is collected at another position, 1 cm away from the stimulation point. See Figure 6.10 for the collected data. It is also observed that the adaptive and uniform results agree very well, almost identical to each other, except that there is slight difference in the upstroke phase but the difference is less than 0.3%. The adaptive computation uses 72.64 seconds while the uniform simulation uses 2160.74 seconds. The speed-up gained by the AMR algorithm is about 29.7.
Figure 6.9: Membrane potential of the Luo-Rudy phase one model in one space dimension (data are collected at $x = 0.5$ cm)
Figure 6.10: Membrane potential of the Beeler-Reuter model in one space dimension (data are collected at $x = 1.0 \text{ cm}$)
As a conclusion of this chapter, we present computational results from the simulation of the Luo-Rudy phase one bidomain model [120] with our AMR algorithm even though the numerical results have not been validated and the experiment uses different modules from those described in section 6.1.

In the simulation, the computational domain is defined by the unit square \([0, 1 \text{ cm}] \times [0, 1 \text{ cm}]\). The fibers are aligned with the positive diagonal. The intracellular conductivity in the principal axis is 6.0 \(mS/cm\); the extracellular conductivity in the principal axis is 24.0 \(mS/cm\). The intracellular conductivity has an anisotropy ratio \(6.0 : 0.6 = 10\); and the extracellular conductivity has an anisotropy ratio \(24.0 : 12.0 = 2\). The membrane capacitance is chosen to be 1.0 \(\mu F/cm^2\). The surface-to-volume ratio \(\beta\) in the bidomain equations (2.1) is set as 4036.5 \(cm^{-1}\).

We fix the extracellular potential at the lower-left corner of the domain to be 0, in order to get a unique solution for the bidomain equations.

The coarsest level grid in the AMR algorithm consists of 16 elements. That is, the unit square domain is initially partitioned into a 4 by 4 grid. Totally, seven AMR levels with refinement ratio two \((r = 2)\) are used in the simulation.

In the experiment, we use a gradient detector for the transmembrane potential \(V_m\), instead of the Richardson extrapolation error estimation procedure, for tagging of cells during the integration process.

In addition, this simulation does not use time interpolation to determine the initial values for linear diffusion on coarse levels. Instead, it first adaptively integrates nonlinear reactions by a half fine time step on all of the coarse nodes that are not covered by fine level grids and then initializes the linear diffusion with the updated values. After the linear diffusion on composite grids is integrated by a full fine time step, the nonlinear reactions on all of those coarse nodes are adaptively integrated by another half fine time step. With these steps, the AMR algorithm for this simulation
is essentially space-adaptive only.

Some snapshots after a stimulus is applied at the center of the domain are shown in Figure 6.11. Those on the left represent the adaptively refined grids and iso-contours of the transmembrane potential $V_m = \Phi_i - \Phi_e$; and the right ones denote the adaptively refined grids and the iso-contours of the extra-cellular potential $\Phi_e$. Only those iso-potentials with values in the range of $-20 \text{ mV}$ to $20 \text{ mV}$ are plotted.

**Remark:** The AMR algorithm has also been applied to solving hyperbolic equations by conservation laws such as the Sod's shock-tube problem (1D), the mach
(c) Iso-contours of transmembrane and extracellular potentials (time = 4.20703 ms)

(d) Iso-contours of transmembrane and extracellular potential (time = 4.60547 ms)
(e) Iso-contours of transmembrane and extracellular potentials (time = 5.00391 ms)

(f) Iso-contours of transmembrane and extracellular potential (time = 5.40625 ms)
(g) Iso-contours of transmembrane and extracellular potentials (time = 6.60547 ms)

(h) Iso-contours of transmembrane and extracellular potentials (time = 7.00391 ms)

**Figure 6.11**: Luo-Rudy phase one bidomain model: transmembrane and extracellular potentials
3 wind tunnel with a step problem (2D), the oblique shock reflection problem (2D) and super-sonic flow around cylinders (2D) etc. See Appendix F for some results of the AMR for the hyperbolic conservation laws of gas dynamics. Dynamic animations of the numerical results, including those for the reaction-diffusion problems, are available on-line at http://www.math.duke.edu/~ying/amr.html.
Table 6.8: AMR for the second 2D test problem at time $t = 0.25$

<table>
<thead>
<tr>
<th>#refinement levels</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>finest cell size</td>
<td>4/64</td>
<td>4/128</td>
<td>4/256</td>
<td>4/512</td>
<td>4/1024</td>
</tr>
<tr>
<td>#finest cells</td>
<td>500</td>
<td>1568</td>
<td>5512</td>
<td>20736</td>
<td>84272</td>
</tr>
<tr>
<td>#multigrid iterations</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>error in $L^2$ norm</td>
<td>0.0420310</td>
<td>0.0113121</td>
<td>0.0026852</td>
<td>0.0006743</td>
<td>0.0001745</td>
</tr>
</tbody>
</table>

Table 6.9: AMR for the second 2D test problem at time $t = 0.5$

<table>
<thead>
<tr>
<th>#refinement levels</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>finest cell size</td>
<td>4/64</td>
<td>4/128</td>
<td>4/256</td>
<td>4/512</td>
<td>4/1024</td>
</tr>
<tr>
<td>#finest cells</td>
<td>608</td>
<td>1896</td>
<td>6580</td>
<td>24992</td>
<td>100908</td>
</tr>
<tr>
<td>#multigrid iterations</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>error in $L^2$ norm</td>
<td>0.0882222</td>
<td>0.0254271</td>
<td>0.0061091</td>
<td>0.0015778</td>
<td>0.0004370</td>
</tr>
</tbody>
</table>

Table 6.10: AMR for the second 2D test problem at time $t = 1$

<table>
<thead>
<tr>
<th>#refinement levels</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>finest cell size</td>
<td>4/64</td>
<td>4/128</td>
<td>4/256</td>
<td>4/512</td>
<td>4/1024</td>
</tr>
<tr>
<td>#finest cells</td>
<td>788</td>
<td>2520</td>
<td>8840</td>
<td>33336</td>
<td>134476</td>
</tr>
<tr>
<td>#multigrid iterations</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>error in $L^2$ norm</td>
<td>0.1982986</td>
<td>0.0589566</td>
<td>0.0143002</td>
<td>0.0037686</td>
<td>0.0010959</td>
</tr>
</tbody>
</table>

Table 6.11: AMR for the second 2D test problem at time $t = 2$

<table>
<thead>
<tr>
<th>#refinement levels</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>finest cell size</td>
<td>4/64</td>
<td>4/128</td>
<td>4/256</td>
<td>4/512</td>
<td>4/1024</td>
</tr>
<tr>
<td>#finest cells</td>
<td>1176</td>
<td>3788</td>
<td>13312</td>
<td>50104</td>
<td>202468</td>
</tr>
<tr>
<td>#multigrid iterations</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>error in $L^2$ norm</td>
<td>0.4523854</td>
<td>0.1442141</td>
<td>0.0352707</td>
<td>0.0093734</td>
<td>0.0027784</td>
</tr>
</tbody>
</table>
Table 6.12: Times of AMR for the 1D FitzHugh-Nagumo problem (tol\textsubscript{AMR} = 0.1)

<table>
<thead>
<tr>
<th>#refinement levels</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>finest cell size</td>
<td>1/64</td>
<td>1/128</td>
<td>1/256</td>
<td>1/512</td>
<td>1/1024</td>
<td>1/2048</td>
</tr>
<tr>
<td>#finest cells</td>
<td>32</td>
<td>48</td>
<td>72</td>
<td>112</td>
<td>144</td>
<td>20</td>
</tr>
<tr>
<td>SDIRK time (secs)</td>
<td>0.85</td>
<td>2.46</td>
<td>7.54</td>
<td>21.99</td>
<td>56.68</td>
<td>77.22</td>
</tr>
<tr>
<td>multigrid time (secs)</td>
<td>0.03</td>
<td>0.16</td>
<td>0.45</td>
<td>1.57</td>
<td>4.95</td>
<td>16.92</td>
</tr>
<tr>
<td>AMR time (secs)</td>
<td>0.04</td>
<td>0.02</td>
<td>0.06</td>
<td>0.24</td>
<td>0.61</td>
<td>1.19</td>
</tr>
<tr>
<td>total time (secs)</td>
<td>1.0</td>
<td>2.77</td>
<td>8.33</td>
<td>24.59</td>
<td>64.61</td>
<td>100.20</td>
</tr>
<tr>
<td>speed-up</td>
<td>1.29</td>
<td>1.81</td>
<td>2.39</td>
<td>3.26</td>
<td>4.93</td>
<td>12.7</td>
</tr>
</tbody>
</table>

Table 6.13: Times of uniform refinement for the 1D FitzHugh-Nagumo problem

<table>
<thead>
<tr>
<th>#refinement levels</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>finest cell size</td>
<td>1/64</td>
<td>1/128</td>
<td>1/256</td>
<td>1/512</td>
<td>1/1024</td>
<td>1/2048</td>
</tr>
<tr>
<td>#finest cells</td>
<td>64</td>
<td>128</td>
<td>256</td>
<td>512</td>
<td>1024</td>
<td>2048</td>
</tr>
<tr>
<td>SDIRK time (secs)</td>
<td>1.27</td>
<td>4.96</td>
<td>19.52</td>
<td>77.97</td>
<td>302.73</td>
<td>1169.45</td>
</tr>
<tr>
<td>multigrid time (secs)</td>
<td>0.01</td>
<td>0.06</td>
<td>0.31</td>
<td>1.76</td>
<td>14.58</td>
<td>101.85</td>
</tr>
<tr>
<td>total time (secs)</td>
<td>1.29</td>
<td>5.03</td>
<td>19.91</td>
<td>80.06</td>
<td>318.50</td>
<td>1276.99</td>
</tr>
</tbody>
</table>
### Table 6.14: Times of AMR for the 1D FitzHugh-Nagumo problem ($tol_{AMR} = 0.02$)

<table>
<thead>
<tr>
<th>#refinement levels</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>finest cell size</td>
<td>1/64</td>
<td>1/128</td>
<td>1/256</td>
<td>1/512</td>
<td>1/1024</td>
<td>1/2048</td>
</tr>
<tr>
<td>#finest cells</td>
<td>40</td>
<td>60</td>
<td>104</td>
<td>160</td>
<td>280</td>
<td>432</td>
</tr>
<tr>
<td>SDIRK time (secs)</td>
<td>1.12</td>
<td>3.30</td>
<td>10.00</td>
<td>31.66</td>
<td>102.75</td>
<td>314.97</td>
</tr>
<tr>
<td>multigrid time (secs)</td>
<td>0.04</td>
<td>0.15</td>
<td>0.55</td>
<td>2.01</td>
<td>6.74</td>
<td>30.81</td>
</tr>
<tr>
<td>AMR time (secs)</td>
<td>0.03</td>
<td>0.04</td>
<td>0.11</td>
<td>0.38</td>
<td>0.86</td>
<td>2.08</td>
</tr>
<tr>
<td>total time (secs)</td>
<td>1.25</td>
<td>3.69</td>
<td>11.10</td>
<td>34.76</td>
<td>112.94</td>
<td>355.28</td>
</tr>
<tr>
<td>speed-up</td>
<td>1.03</td>
<td>1.36</td>
<td>1.79</td>
<td>2.30</td>
<td>2.82</td>
<td>3.59</td>
</tr>
</tbody>
</table>

### Table 6.15: Times of AMR for the 1D FitzHugh-Nagumo problem ($tol_{AMR} = 0.05$)

<table>
<thead>
<tr>
<th>#refinement levels</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>finest cell size</td>
<td>1/64</td>
<td>1/128</td>
<td>1/256</td>
<td>1/512</td>
<td>1/1024</td>
<td>1/2048</td>
</tr>
<tr>
<td>#finest cells</td>
<td>36</td>
<td>52</td>
<td>88</td>
<td>136</td>
<td>208</td>
<td>276</td>
</tr>
<tr>
<td>SDIRK time (secs)</td>
<td>0.99</td>
<td>2.87</td>
<td>8.74</td>
<td>26.48</td>
<td>80.70</td>
<td>209.89</td>
</tr>
<tr>
<td>multigrid time (secs)</td>
<td>0.03</td>
<td>0.11</td>
<td>0.51</td>
<td>1.66</td>
<td>5.68</td>
<td>23.0</td>
</tr>
<tr>
<td>AMR time (secs)</td>
<td>0.01</td>
<td>0.04</td>
<td>0.16</td>
<td>0.27</td>
<td>0.71</td>
<td>1.78</td>
</tr>
<tr>
<td>total time (secs)</td>
<td>1.09</td>
<td>3.17</td>
<td>9.81</td>
<td>29.12</td>
<td>89.31</td>
<td>240.98</td>
</tr>
<tr>
<td>speed-up</td>
<td>1.18</td>
<td>1.59</td>
<td>2.03</td>
<td>2.75</td>
<td>3.57</td>
<td>5.30</td>
</tr>
</tbody>
</table>
Table 6.16: Times of AMR for the 2D FitzHugh-Nagumo problem ($tol_{AMR} = 0.1$)

<table>
<thead>
<tr>
<th>#refinement levels</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>finest cell size</td>
<td>1/32</td>
<td>1/64</td>
<td>1/128</td>
<td>1/256</td>
</tr>
<tr>
<td>SDIRK time (secs)</td>
<td>17.87</td>
<td>100.04</td>
<td>565.08</td>
<td>3353.77</td>
</tr>
<tr>
<td>multigrid time (secs)</td>
<td>0.68</td>
<td>7.53</td>
<td>72.06</td>
<td>549.85</td>
</tr>
<tr>
<td>AMR time (secs)</td>
<td>0.24</td>
<td>1.16</td>
<td>7.54</td>
<td>49.12</td>
</tr>
<tr>
<td>total time (secs)</td>
<td>19.07</td>
<td>110.44</td>
<td>655.02</td>
<td>4019.25</td>
</tr>
<tr>
<td>speed-up</td>
<td>1.40</td>
<td>1.98</td>
<td>2.75</td>
<td>3.46</td>
</tr>
</tbody>
</table>

Table 6.17: Times of uniform refinement for the 2D FitzHugh-Nagumo problem

<table>
<thead>
<tr>
<th>#refinement levels</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>finest cell size</td>
<td>1/32</td>
<td>1/64</td>
<td>1/128</td>
<td>1/256</td>
</tr>
<tr>
<td>#finest cells</td>
<td>1024</td>
<td>4096</td>
<td>16384</td>
<td>65536</td>
</tr>
<tr>
<td>SDIRK time (secs)</td>
<td>25.35</td>
<td>198.19</td>
<td>1518.36</td>
<td>11562.03</td>
</tr>
<tr>
<td>multigrid time (secs)</td>
<td>1.05</td>
<td>17.98</td>
<td>257.24</td>
<td>2124.24</td>
</tr>
<tr>
<td>total time (secs)</td>
<td>26.73</td>
<td>219.17</td>
<td>1802.46</td>
<td>13899.56</td>
</tr>
</tbody>
</table>
Chapter 7

Conclusion and Future Work

In this chapter, we conclude the thesis with a few conclusions and the main achievements are summarized. We also point out some possible future work and improvement of the AMR algorithm to the cardiac problems. They include efficient parallelization of the algorithm, conformal refinement with triangular and tetrahedral elements, incorporation of higher order methods, and even a fully implicit discretization of the problem with nonlinear multigrid solver.

7.1 Thesis Achievements

In the thesis work, a space-time adaptive mesh refinement algorithm for the singularly perturbed reaction-diffusion equations, arising from numerical modeling of the electrical wave propagation in the heart has been proposed and implemented. The algorithm first uses a second-order operator splitting to separate the linear diffusion term from the stiff nonlinear reactions in the equations. The stiff nonlinear reactions are adaptively integrated with a second-order singly diagonally implicit Runge-Kutta (SDIRK) method, which is stiffly accurate and absolutely stable (both L- and A-stable). The decoupled linear diffusion is discretized with a second-order conforming finite element method on the adaptively refined grids. The resulting composite grid equations are solved by the standard multigrid iteration method.

The AMR algorithm proposed in the thesis work is aimed to be geometrically flexible as much as unstructured grids based strategies, as easy as structured grids based methods to be implemented, and as efficient as Berger-Oliger’s approach in time stepping. The AMR algorithm is designed and implemented in a unified dimension
independent way. A unified code enables us debug the program in a lower space dimension and then naturally extend to higher space dimensions.

Specifically, it uses quadrilateral or hexahedral elements to construct a hierarchy of properly nested level grids. The grid on each level is generated through regular bisection or refinement from a subgrid of that on the previous coarser level. We represent a grid by lists of elements and the lower dimensional facets, including nodes, edges, and sides. With this grid representation, the connectivity of grid entities is well established, which makes grid-based operations easy to implement. The list based data structure allows us to choose the mesh refinement ratio to be any integer number without imposing unnecessary computer memory overheads.

The AMR algorithm follows Berger-Oliger’s approach [24, 27] in time stepping but not in grid organization. The hierarchical system is recursively integrated. Starting from the root level, a coarse level is first integrated with a large time step, followed by integrating its next finer level with a few small time steps until both levels are synchronized. At the moment of synchronization, the algorithm invokes some of the typical routines such as mesh regridding and data up-/downscaling. As an essential part of an adaptive algorithm, an error estimation technique of Richardson extrapolation type is employed for general partial differential equations.

7.2 Future Work

For more accurate and efficient modeling of electrical wave propagation in the realistic heart, there are still many open problems.

First, a higher-order discretization and implementation of AMR for the cardiac model problems will be studied. Currently, the AMR algorithm uses a second-order conforming finite element method for spatial discretization and a second-order singly diagonally implicit Runge-Kutta (SDIRK) method for temporal discretization. To
Furthermore reduce the computer memory usage in three-dimensional simulations, it is desirable for the AMR to use less nodes to achieve greater accuracy and efficiency. Piece-wise quadratic or cubic elements can give third or fourth order spatial accuracy while using much less grid nodes. Due to the unavailability of high order (greater than two) stable operator splitting techniques for the singularly perturbed reaction-diffusion equations, an unsplit method must be alternatively investigated to achieve a higher order discretization. At the same time, to achieve higher-order finite element approximation, triangular (2D) or tetrahedral (3D) elements may also be introduced along the coarse-fine interfaces to eliminate hanging nodes. In this phase, an efficient preconditioner for the linear or nonlinear systems resulting from higher order discretizations is also a difficult point to be tackled.

Second, parallelization of the AMR algorithm will definitely speed up the simulation of electrical wave propagation in the whole heart. It is usually a difficult task to parallelize an AMR algorithm due to the frequent grid refinement and node redistribution. Hopefully, the AMR algorithm may perform well in distributed computing using space filling curves to partition a grid into load balanced sub-grids while keeping the communication overhead minimized.

Up to now, the AMR algorithm belongs to the h-refinement category. The third step in the development of the AMR algorithm will involve the combinations of both h-refinement and p-refinement strategies. The hp-adaptive implementation can give much better and even exponential convergence performance.

In the long run, the AMR algorithm must take into account the heart contraction during simulations. To deal with moving boundaries, we will study on the incorporation of Cartesian grid methods, phase-field methods or fictitious domain methods, which are commonly used for problems on complex domains. In the future, we are also interested in combinations of AMR and the recently proposed meshfree methods.
Appendix A

Results by Space Only Adaptive Integration

The numerical results presented here are also from experiments with our AMR algorithm for the test problems presented in section 6.2. They correspondingly use the same tolerances, parameters and modules as those in section 6.2 except that, instead of using time interpolation to determine the initial values for linear diffusion on coarse levels, the algorithm first adaptively integrates nonlinear reactions by a half fine time step on all of the coarse nodes that are not covered by fine level grids and then initializes the linear diffusion with the updated values. After the linear diffusion on composite grids is integrated by a full fine time step, the nonlinear reactions on all of those coarse nodes are adaptively integrated by another half fine time step. These steps exactly follow the Strang splitting procedure. The resulting integration is essentially a space only adaptive mesh refinement process for the reaction-diffusion problems.

A.1 The Case in One Space Dimension

Example A.1: Let us consider the following scalar singularly perturbed reaction-diffusion equation

\[
\frac{\partial u}{\partial t} = \epsilon \Delta u + \frac{2}{\epsilon} (1 - u)u^2 \quad \text{for } t > 0, \ x \in (0, 4),
\]

with \(\epsilon = 1/32\). The initial values are given by

\[
u(t, x) = \frac{1}{1 + e^{(x-t-1.5)/\epsilon}}.
\]
Homogeneous Neumann boundary conditions are applied.

**Table A.1:** Space Only AMR for the 1D test at time $t = 0.25$

<table>
<thead>
<tr>
<th>#refinement levels</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>#finest cells</td>
<td>26</td>
<td>44</td>
<td>88</td>
<td>176</td>
<td>348</td>
</tr>
<tr>
<td>#multigrid iterations</td>
<td>2</td>
<td>3</td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>error in $L^2$ norm</td>
<td>0.0065727</td>
<td>0.0014701</td>
<td>0.0003702</td>
<td>0.0000926</td>
<td>0.0000232</td>
</tr>
</tbody>
</table>

**Table A.2:** Space Only AMR for the 1D test at time $t = 0.5$

<table>
<thead>
<tr>
<th>#refinement levels</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>#finest cells</td>
<td>26</td>
<td>44</td>
<td>88</td>
<td>176</td>
<td>348</td>
</tr>
<tr>
<td>#multigrid iterations</td>
<td>2</td>
<td>3</td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>error in $L^2$ norm</td>
<td>0.0141842</td>
<td>0.0031724</td>
<td>0.0008046</td>
<td>0.0002026</td>
<td>0.0000525</td>
</tr>
</tbody>
</table>

**Table A.3:** Space Only for the 1D test at time $t = 1$

<table>
<thead>
<tr>
<th>#refinement levels</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>#finest cells</td>
<td>26</td>
<td>46</td>
<td>88</td>
<td>176</td>
<td>348</td>
</tr>
<tr>
<td>#multigrid iterations</td>
<td>2</td>
<td>3</td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>error in $L^2$ norm</td>
<td>0.0291475</td>
<td>0.0065517</td>
<td>0.0016688</td>
<td>0.0004228</td>
<td>0.0001128</td>
</tr>
</tbody>
</table>

The problem is solved on the interval $[0, 4]$ in space from 0 to 1 in time with the spatial only AMR algorithm. Numerical results are shown in Tables A.1-A.3.

**A.2 The Cases in Two Space Dimensions**

**Example A.2:** Let us again consider the scalar reaction-diffusion equation

$$\frac{\partial u}{\partial t} = \epsilon \Delta u + \frac{2}{\epsilon} (1 - u)u^2 \quad \text{for } t > 0, \; x \in (0, 4)^2; \quad (A.1)$$

with $\epsilon = 1/32$. The initial values are given by

$$u(t, x) \equiv u(t, x_1, x_2) = \frac{1}{1 + e^{(x_1-t-1.5)/\epsilon}},$$

211
and homogeneous Neumann boundary conditions are applied.

**Table A.4:** Space Only AMR for the first 2D test at time $t = 0.25$

<table>
<thead>
<tr>
<th>#refinement levels</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>finest cell size</td>
<td>4/64</td>
<td>4/128</td>
<td>4/256</td>
<td>4/512</td>
<td>4/1024</td>
</tr>
<tr>
<td>#finest cells</td>
<td>1024</td>
<td>3072</td>
<td>10752</td>
<td>40960</td>
<td>167936</td>
</tr>
<tr>
<td>#multigrid iterations</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>error in $L^2$ norm</td>
<td>0.0401401</td>
<td>0.0128398</td>
<td>0.0029790</td>
<td>0.0007447</td>
<td>0.0001895</td>
</tr>
</tbody>
</table>

**Table A.5:** Space Only AMR for the first 2D test at time $t = 0.5$

<table>
<thead>
<tr>
<th>#refinement levels</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>finest cell size</td>
<td>4/64</td>
<td>4/128</td>
<td>4/256</td>
<td>4/512</td>
<td>4/1024</td>
</tr>
<tr>
<td>#finest cells</td>
<td>1024</td>
<td>3072</td>
<td>10752</td>
<td>40960</td>
<td>167936</td>
</tr>
<tr>
<td>#multigrid iterations</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>error in $L^2$ norm</td>
<td>0.0761273</td>
<td>0.0275317</td>
<td>0.0064490</td>
<td>0.0016276</td>
<td>0.0004223</td>
</tr>
</tbody>
</table>

**Table A.6:** Space Only AMR for the first 2D test at time $t = 1$

<table>
<thead>
<tr>
<th>#refinement levels</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>finest cell size</td>
<td>4/64</td>
<td>4/128</td>
<td>4/256</td>
<td>4/512</td>
<td>4/1024</td>
</tr>
<tr>
<td>#finest cells</td>
<td>1024</td>
<td>3072</td>
<td>10752</td>
<td>40960</td>
<td>167936</td>
</tr>
<tr>
<td>#multigrid iterations</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>error in $L^2$ norm</td>
<td>0.1781283</td>
<td>0.0563727</td>
<td>0.0133443</td>
<td>0.0033894</td>
<td>0.0008927</td>
</tr>
</tbody>
</table>

The problem is solved on the square rectangular domain $[0, 4] \times [0, 4]$ from 0 to 1 in time. Numerical results are shown in Tables A.4-A.6. The coarsest level grid has $16 \times 16 = 256$ elements of equal size with $\Delta x = 0.25$.

**Example A.3.** Let us consider the following nonlinear reaction-diffusion equation:

$$\frac{\partial u}{\partial t} = \epsilon \Delta u + \frac{1}{\epsilon} u(1 - u) \left\{2u + \frac{\epsilon}{|x|}\right\} \quad \text{for } t > 0, \; x \in (0, 4)^2, \quad (A.2)$$

with $\epsilon = 1/32$. The initial values are given by

$$u = \frac{1}{1 + e^{(|x| - t - 1)/\epsilon}}.$$
and homogeneous Neumann boundary conditions are applied. The problem is solved by the space adaptive only algorithm in the domain \((0, 4)^2\) from 0 to 2 in time. Numerical results are shown in Tables A.7-A.9. The coarsest grid has \(16 \times 16\) elements. The time step size is always chosen to be equal to the grid size on each level.

**Table A.7:** Space Only AMR for the second 2D test at time \(t = 0.25\)

<table>
<thead>
<tr>
<th>#refinement levels</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>finest cell size</td>
<td>4/64</td>
<td>4/128</td>
<td>4/256</td>
<td>4/512</td>
<td>4/1024</td>
</tr>
<tr>
<td>#finest cells</td>
<td>500</td>
<td>1568</td>
<td>5520</td>
<td>20736</td>
<td>83632</td>
</tr>
<tr>
<td>#multigrid iterations</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>error in (L^2) norm</td>
<td>0.0420310</td>
<td>0.0113121</td>
<td>0.0026837</td>
<td>0.0006696</td>
<td>0.0001696</td>
</tr>
</tbody>
</table>

**Table A.8:** Space Only AMR for the second 2D test at time \(t = 0.5\)

<table>
<thead>
<tr>
<th>#refinement levels</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>finest cell size</td>
<td>4/64</td>
<td>4/128</td>
<td>4/256</td>
<td>4/512</td>
<td>4/1024</td>
</tr>
<tr>
<td>#finest cells</td>
<td>608</td>
<td>1896</td>
<td>6580</td>
<td>24976</td>
<td>100304</td>
</tr>
<tr>
<td>#multigrid iterations</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>error in (L^2) norm</td>
<td>0.0882242</td>
<td>0.0254200</td>
<td>0.0060833</td>
<td>0.0015301</td>
<td>0.0003939</td>
</tr>
</tbody>
</table>

**Table A.9:** Space Only AMR for the second 2D test at time \(t = 1\)

<table>
<thead>
<tr>
<th>#refinement levels</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>finest cell size</td>
<td>4/64</td>
<td>4/128</td>
<td>4/256</td>
<td>4/512</td>
<td>4/1024</td>
</tr>
<tr>
<td>#finest cells</td>
<td>788</td>
<td>2520</td>
<td>8828</td>
<td>33320</td>
<td>134064</td>
</tr>
<tr>
<td>#multigrid iterations</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>error in (L^2) norm</td>
<td>0.1983090</td>
<td>0.0589074</td>
<td>0.0141886</td>
<td>0.0035888</td>
<td>0.0009356</td>
</tr>
</tbody>
</table>

**Table A.10:** Space Only AMR for the second 2D test at time \(t = 2\)

<table>
<thead>
<tr>
<th>#refinement levels</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>finest cell size</td>
<td>4/64</td>
<td>4/128</td>
<td>4/256</td>
<td>4/512</td>
<td>4/1024</td>
</tr>
<tr>
<td>#finest cells</td>
<td>1176</td>
<td>3788</td>
<td>13296</td>
<td>50140</td>
<td>201544</td>
</tr>
<tr>
<td>#multigrid iterations</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>error in (L^2) norm</td>
<td>0.4519031</td>
<td>0.1440550</td>
<td>0.0349441</td>
<td>0.0088635</td>
<td>0.0023227</td>
</tr>
</tbody>
</table>
Appendix B

Results with Exact Error Based Tagging of Cells

The numerical results presented here are also from experiments with our AMR algorithm for the test problems presented in section 6.2. They correspondingly use the same tolerances, parameters and modules (space only adaptive integration) as those in Appendix A except that tagging of cells in these experiments uses exact errors since the exact solutions are known for the test problems, instead of using the Richardson extrapolation based error estimation procedure. By the results presented in this appendix, we demonstrate that the “overly” refined regions observed in section 6.2 are not because of the use of linear interpolation to obtain the monitored data \( w^c(\nu \Delta t) \). In the experiments, a cell is tagged if the exact relative error is greater than the specific tolerance, i.e.,

\[
|e| \cdot \frac{\mathcal{L}}{\lambda \cdot \Delta t} > tol_{AMR},
\]

where \( e \) is the exact error of fine data and the tolerance \( tol_{AMR} \) is defined in (6.2). In the test problems, the time length scale \( \mathcal{L} \) is set to be one. Each of the exact traveling wave solutions has maximum value one and wave speed \( \lambda = 1 \).

An interesting phenomenon observed is that the data from these experiments even converges slower than that from those using the Richardson extrapolation based error estimation since the process using exact errors tagges less cells for refinement and less computational efforts gain less numerical accuracy.
B.1 The Case in One Space Dimension

Example B.1: Let us consider the following scalar singularly perturbed reaction-diffusion equation

$$\frac{\partial u}{\partial t} = \epsilon \Delta u + \frac{2}{\epsilon}(1 - u)u^2 \quad \text{for } t > 0, \ x \in (0, 4),$$

with $\epsilon = 1/32$. The initial values are given by

$$u(t, x) = \frac{1}{1 + e^{(x-t-1.5)/\epsilon}}.$$

Homogeneous Neumann boundary conditions are applied.

### Table B.1: Space Only AMR for the 1D test at time $t = 0.25$

<table>
<thead>
<tr>
<th>#refinement levels</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>#finest cells</td>
<td>24</td>
<td>40</td>
<td>76</td>
<td>148</td>
<td>292</td>
</tr>
<tr>
<td>#multigrid iterations</td>
<td>2</td>
<td>3</td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>error in $L^2$ norm</td>
<td>0.0065732</td>
<td>0.0014721</td>
<td>0.0003720</td>
<td>0.0000959</td>
<td>0.0000272</td>
</tr>
</tbody>
</table>

### Table B.2: Space Only AMR for the 1D test at time $t = 0.5$

<table>
<thead>
<tr>
<th>#refinement levels</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>#finest cells</td>
<td>24</td>
<td>40</td>
<td>76</td>
<td>148</td>
<td>292</td>
</tr>
<tr>
<td>#multigrid iterations</td>
<td>2</td>
<td>3</td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>error in $L^2$ norm</td>
<td>0.0141910</td>
<td>0.0031877</td>
<td>0.0008199</td>
<td>0.0002242</td>
<td>0.0000732</td>
</tr>
</tbody>
</table>

### Table B.3: Space Only for the 1D test at time $t = 1$

<table>
<thead>
<tr>
<th>#refinement levels</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>#finest cells</td>
<td>24</td>
<td>42</td>
<td>78</td>
<td>148</td>
<td>292</td>
</tr>
<tr>
<td>#multigrid iterations</td>
<td>2</td>
<td>3</td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>error in $L^2$ norm</td>
<td>0.0291701</td>
<td>0.0065991</td>
<td>0.0017269</td>
<td>0.0004796</td>
<td>0.0001780</td>
</tr>
</tbody>
</table>

The problem is solved on the interval $[0, 4]$ in space from 0 to 1 in time with the spatial only AMR algorithm. Numerical results are shown in Tables B.1-B.3.
B.2 The Cases in Two Space Dimensions

Example B.2: Let us again consider the scalar reaction-diffusion equation

\[ \frac{\partial u}{\partial t} = \epsilon \Delta u + \frac{2}{\epsilon} (1 - u) u^2 \quad \text{for } t > 0, \; x \in (0, 4)^2, \]  

with \( \epsilon = 1/32 \). The initial values are given by

\[ u(t, x) \equiv u(t, x_1, x_2) = \frac{1}{1 + e^{(x_1 - t - 1.5)/\epsilon}}, \]

and homogeneous Neumann boundary conditions are applied.

Table B.4: Space Only AMR for the first 2D test at time \( t = 0.25 \)

<table>
<thead>
<tr>
<th>#refinement levels</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>finest cell size</td>
<td>4/64</td>
<td>4/128</td>
<td>4/256</td>
<td>4/512</td>
<td>4/1024</td>
</tr>
<tr>
<td>#finest cells</td>
<td>768</td>
<td>2560</td>
<td>10240</td>
<td>35936</td>
<td>145456</td>
</tr>
<tr>
<td>#multigrid iterations</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>error in ( L^2 ) norm</td>
<td>0.0401381</td>
<td>0.0128595</td>
<td>0.0029959</td>
<td>0.0007588</td>
<td>0.0002021</td>
</tr>
</tbody>
</table>

Table B.5: Space Only AMR for the first 2D test at time \( t = 0.5 \)

<table>
<thead>
<tr>
<th>#refinement levels</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>finest cell size</td>
<td>4/64</td>
<td>4/128</td>
<td>4/256</td>
<td>4/512</td>
<td>4/1024</td>
</tr>
<tr>
<td>#finest cells</td>
<td>768</td>
<td>2560</td>
<td>10240</td>
<td>35704</td>
<td>147456</td>
</tr>
<tr>
<td>#multigrid iterations</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>error in ( L^2 ) norm</td>
<td>0.0763714</td>
<td>0.0276768</td>
<td>0.0065078</td>
<td>0.0016997</td>
<td>0.0005479</td>
</tr>
</tbody>
</table>

Table B.6: Space Only AMR for the first 2D test at time \( t = 1 \)

<table>
<thead>
<tr>
<th>#refinement levels</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>finest cell size</td>
<td>4/64</td>
<td>4/128</td>
<td>4/256</td>
<td>4/512</td>
<td>4/1024</td>
</tr>
<tr>
<td>#finest cells</td>
<td>512</td>
<td>2304</td>
<td>9728</td>
<td>35388</td>
<td>147456</td>
</tr>
<tr>
<td>#multigrid iterations</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>error in ( L^2 ) norm</td>
<td>0.1831447</td>
<td>0.0567898</td>
<td>0.0134644</td>
<td>0.0037639</td>
<td>0.0011316</td>
</tr>
</tbody>
</table>

The problem is solved on the square rectangular domain \([0, 4] \times [0, 4]\) from 0 to 1 in time. Numerical results are shown in Tables B.4-B.6. The coarsest level grid has \( 16 \times 16 = 256 \) elements of equal size with \( \Delta x = 0.25 \).
Example B.3. Let us consider the following nonlinear reaction-diffusion equation:

\[
\frac{\partial u}{\partial t} = \epsilon \Delta u + \frac{1}{\epsilon} u(1 - u) \left\{ 2u + \frac{\epsilon}{|x|} \right\} \quad \text{for } t > 0, \; x \in (0, 4)^2, \quad (B.3)
\]

with \( \epsilon = 1/32 \). The initial values are given by

\[
u = \frac{1}{1 + e^{(|x| - t - 1)/\epsilon}},
\]

and homogeneous Neumann boundary conditions are applied. The problem is solved by the space adaptive only algorithm in the domain \((0, 4)^2\) from 0 to 2 in time. Numerical results are shown in Tables B.7-B.9. The coarsest grid has \(16 \times 16\) elements. The time step size is always chosen to be equal to the grid size on each level.
### Table B.7: Space Only AMR for the second 2D test at time $t = 0.25$

<table>
<thead>
<tr>
<th>#refinement levels</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>finest cell size</td>
<td>4/64</td>
<td>4/128</td>
<td>4/256</td>
<td>4/512</td>
<td>4/1024</td>
</tr>
<tr>
<td>#finest cells</td>
<td>380</td>
<td>1372</td>
<td>5004</td>
<td>17796</td>
<td>71756</td>
</tr>
<tr>
<td>#multigrid iterations</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>error in $L^2$ norm</td>
<td>0.0420318</td>
<td>0.0113173</td>
<td>0.0026903</td>
<td>0.0006796</td>
<td>0.0001784</td>
</tr>
</tbody>
</table>

### Table B.8: Space Only AMR for the second 2D test at time $t = 0.5$

<table>
<thead>
<tr>
<th>#refinement levels</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>finest cell size</td>
<td>4/64</td>
<td>4/128</td>
<td>4/256</td>
<td>4/512</td>
<td>4/1024</td>
</tr>
<tr>
<td>#finest cells</td>
<td>452</td>
<td>1684</td>
<td>6036</td>
<td>21404</td>
<td>86248</td>
</tr>
<tr>
<td>#multigrid iterations</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>error in $L^2$ norm</td>
<td>0.0882981</td>
<td>0.0254683</td>
<td>0.0061206</td>
<td>0.0016005</td>
<td>0.0004440</td>
</tr>
</tbody>
</table>

### Table B.9: Space Only AMR for the second 2D test at time $t = 1$

<table>
<thead>
<tr>
<th>#refinement levels</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>finest cell size</td>
<td>4/64</td>
<td>4/128</td>
<td>4/256</td>
<td>4/512</td>
<td>4/1024</td>
</tr>
<tr>
<td>#finest cells</td>
<td>1024</td>
<td>3232</td>
<td>11360</td>
<td>46716</td>
<td>173516</td>
</tr>
<tr>
<td>#multigrid iterations</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>error in $L^2$ norm</td>
<td>0.4533199</td>
<td>0.1452271</td>
<td>0.0357126</td>
<td>0.0096792</td>
<td>0.0029646</td>
</tr>
</tbody>
</table>

### Table B.10: Space Only AMR for the second 2D test at time $t = 2$

<table>
<thead>
<tr>
<th>#refinement levels</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>finest cell size</td>
<td>4/64</td>
<td>4/128</td>
<td>4/256</td>
<td>4/512</td>
<td>4/1024</td>
</tr>
<tr>
<td>#finest cells</td>
<td>1176</td>
<td>3788</td>
<td>13296</td>
<td>50140</td>
<td>201544</td>
</tr>
<tr>
<td>#multigrid iterations</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>error in $L^2$ norm</td>
<td>0.4519031</td>
<td>0.1440550</td>
<td>0.0349441</td>
<td>0.0088635</td>
<td>0.0023227</td>
</tr>
</tbody>
</table>
Appendix C

Results with Larger Tolerance for Tagging of Cells

The numerical results presented here are also from experiments with our AMR algorithm for the test problems presented in section 6.2. They correspondingly use the same tolerances, parameters and modules (space only adaptive integration) as those in Appendix B except that, the tolerance for tagging of cells is three times larger. That is, in the experiments presented next, a cell is tagged if the exact relative error is greater than the larger tolerance, i.e.,

\[ |e| \cdot \frac{\mathcal{L}}{\lambda \cdot \Delta t} > 4 \cdot tol_{AMR}, \]  

(C.1)

where \( e \) is the exact error of fine data and the tolerance \( tol_{AMR} \) is defined in (6.2). In the test problems, the time length scale \( \mathcal{L} \) is set to be one. Each of the exact traveling wave solutions has maximum value one and wave speed \( \lambda = 1 \).

By numerical results, we demonstrate that the tolerance for tagging of cells is stringent even with the exact error based strategy to show that the convergence rate is second-order in \( L^2 \) norm. This further indicates that the “overly” refined regions on fine levels observed in section 6.2 are neither because of the use of linear interpolation to obtain the monitored data \( w^e(\nu \Delta t) \) nor because of the small tolerance \( tol_{AMR} \) defined in (6.2). Note that, when too many levels of mesh refinement are used, the un-controlled error from coarse levels (probably together with the roundoff error) dominates in the 1D test case, since the tolerance is so big, three times larger than before. In this case, it seems that the fine preference assumption for our algorithm is violated.
C.1 The Case in One Space Dimension

Example C.1: Let us consider the following scalar singularly perturbed reaction-diffusion equation
\[ \frac{\partial u}{\partial t} = \epsilon \Delta u + \frac{2}{\epsilon} (1 - u) u^2 \quad \text{for } t > 0, \ x \in (0, 4), \]
with \( \epsilon = 1/32 \). The initial values are given by
\[ u(t, x) = \frac{1}{1 + e^{(x-t-1.5)/\epsilon}}. \]
Homogeneous Neumann boundary conditions are applied.

Table C.1: Space Only AMR for the 1D test at time \( t = 0.25 \)

<table>
<thead>
<tr>
<th>#refinement levels</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>#finest cells</td>
<td>14</td>
<td>28</td>
<td>52</td>
<td>100</td>
<td>196</td>
</tr>
<tr>
<td>#multigrid iterations</td>
<td>2</td>
<td>3</td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>error in ( L^2 ) norm</td>
<td>0.0065789</td>
<td>0.0015042</td>
<td>0.0004386</td>
<td>0.0002620</td>
<td>0.0002610</td>
</tr>
</tbody>
</table>

Table C.2: Space Only AMR for the 1D test at time \( t = 0.5 \)

<table>
<thead>
<tr>
<th>#refinement levels</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>#finest cells</td>
<td>18</td>
<td>28</td>
<td>52</td>
<td>100</td>
<td>196</td>
</tr>
<tr>
<td>#multigrid iterations</td>
<td>2</td>
<td>3</td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>error in ( L^2 ) norm</td>
<td>0.0144195</td>
<td>0.0035417</td>
<td>0.0013529</td>
<td>0.0008819</td>
<td>0.0010954</td>
</tr>
</tbody>
</table>

Table C.3: Space Only for the 1D test at time \( t = 1 \)

<table>
<thead>
<tr>
<th>#refinement levels</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>#finest cells</td>
<td>18</td>
<td>30</td>
<td>52</td>
<td>100</td>
<td>196</td>
</tr>
<tr>
<td>#multigrid iterations</td>
<td>2</td>
<td>3</td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>error in ( L^2 ) norm</td>
<td>0.0299849</td>
<td>0.0080246</td>
<td>0.0033126</td>
<td>0.0022902</td>
<td>0.0026832</td>
</tr>
</tbody>
</table>

The problem is solved on the interval \([0, 4]\) in space from 0 to 1 in time with the space only AMR algorithm. Numerical results are shown in Tables C.1-C.3.
C.2 The Cases in Two Space Dimensions

Example C.2: Let us again consider the scalar reaction-diffusion equation

\[
\frac{\partial u}{\partial t} = \epsilon \Delta u + \frac{2}{\epsilon}(1 - u)u^2 \quad \text{for } t > 0, \ x \in (0, 4)^2, \tag{C.2}
\]

with \( \epsilon = 1/32 \). The initial values are given by

\[ u(t, x) \equiv u(t, x_1, x_2) = \frac{1}{1 + e^{(x_1-t-1.5)/\epsilon}}, \]

and homogeneous Neumann boundary conditions are applied.

Table C.4: Space Only AMR for the first 2D test at time \( t = 0.25 \)

<table>
<thead>
<tr>
<th>#refinement levels</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>finest cell size</td>
<td>4/64</td>
<td>4/128</td>
<td>4/256</td>
<td>4/512</td>
<td>4/1024</td>
</tr>
<tr>
<td>#finest cells</td>
<td>512</td>
<td>1536</td>
<td>7168</td>
<td>24576</td>
<td>98416</td>
</tr>
<tr>
<td>#multigrid iterations</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>error in ( L^2 ) norm</td>
<td>0.0405908</td>
<td>0.0148966</td>
<td>0.0039320</td>
<td>0.0016619</td>
<td>0.0014277</td>
</tr>
</tbody>
</table>

Table C.5: Space Only AMR for the first 2D test at time \( t = 0.5 \)

<table>
<thead>
<tr>
<th>#refinement levels</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>finest cell size</td>
<td>4/64</td>
<td>4/128</td>
<td>4/256</td>
<td>4/512</td>
<td>4/1024</td>
</tr>
<tr>
<td>#finest cells</td>
<td>512</td>
<td>1808</td>
<td>7168</td>
<td>26616</td>
<td>102400</td>
</tr>
<tr>
<td>#multigrid iterations</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>error in ( L^2 ) norm</td>
<td>0.0813286</td>
<td>0.0334864</td>
<td>0.0094489</td>
<td>0.0044972</td>
<td>0.0037622</td>
</tr>
</tbody>
</table>

Table C.6: Space Only AMR for the first 2D test at time \( t = 1 \)

<table>
<thead>
<tr>
<th>#refinement levels</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>finest cell size</td>
<td>4/64</td>
<td>4/128</td>
<td>4/256</td>
<td>4/512</td>
<td>4/1024</td>
</tr>
<tr>
<td>#finest cells</td>
<td>512</td>
<td>2304</td>
<td>7168</td>
<td>26624</td>
<td>102400</td>
</tr>
<tr>
<td>#multigrid iterations</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>error in ( L^2 ) norm</td>
<td>0.1953879</td>
<td>0.064851</td>
<td>0.0194413</td>
<td>0.0097921</td>
<td>0.0081938</td>
</tr>
</tbody>
</table>

The problem is solved on the square rectangular domain \([0, 4] \times [0, 4]\) from 0 to 1 in time. The numerical results are shown in Tables C.4-C.6. The coarsest level grid has \( 16 \times 16 = 256 \) elements of equal size with \( \Delta x = 0.25 \).
**Example C.3.** Let us consider the following nonlinear reaction-diffusion equation:

\[
\frac{\partial u}{\partial t} = \epsilon \Delta u + \frac{1}{\epsilon} u(1 - u) \left\{ 2u + \frac{\epsilon}{|x|} \right\} \quad \text{for } t > 0, \ x \in (0, 4)^2,
\]

(C.3)

with \( \epsilon = 1/32 \). The initial values are given by

\[
u = \frac{1}{1 + e^{(|x| - t - 1)/\epsilon}},
\]

and homogeneous Neumann boundary conditions are applied. The problem is solved by the space adaptive only algorithm in the domain \((0, 4)^2\) from 0 to 2 in time. Numerical results are shown in Tables C.7-C.9. The coarsest grid has 16×16 elements, the time step size is always chosen to be equal to the grid size on each level.
### Table C.7: Space Only AMR for the second 2D test at time $t = 0.25$

<table>
<thead>
<tr>
<th>#refinement levels</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>finest cell size</td>
<td>4/64</td>
<td>4/128</td>
<td>4/256</td>
<td>4/512</td>
<td>4/1024</td>
</tr>
<tr>
<td>#finest cells</td>
<td>296</td>
<td>944</td>
<td>3944</td>
<td>13996</td>
<td>56816</td>
</tr>
<tr>
<td>#multigrid iterations</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>error in $L^2$ norm</td>
<td>0.0421103</td>
<td>0.0116540</td>
<td>0.0028013</td>
<td>0.0009144</td>
<td>0.0004943</td>
</tr>
</tbody>
</table>

### Table C.8: Space Only AMR for the second 2D test at time $t = 0.5$

<table>
<thead>
<tr>
<th>#refinement levels</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>finest cell size</td>
<td>4/64</td>
<td>4/128</td>
<td>4/256</td>
<td>4/512</td>
<td>4/1024</td>
</tr>
<tr>
<td>#finest cells</td>
<td>368</td>
<td>1204</td>
<td>4656</td>
<td>17460</td>
<td>68224</td>
</tr>
<tr>
<td>#multigrid iterations</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>error in $L^2$ norm</td>
<td>0.0903599</td>
<td>0.0270480</td>
<td>0.0067413</td>
<td>0.0025987</td>
<td>0.0015520</td>
</tr>
</tbody>
</table>

### Table C.9: Space Only AMR for the second 2D test at time $t = 1$

<table>
<thead>
<tr>
<th>#refinement levels</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>finest cell size</td>
<td>4/64</td>
<td>4/128</td>
<td>4/256</td>
<td>4/512</td>
<td>4/1024</td>
</tr>
<tr>
<td>#finest cells</td>
<td>532</td>
<td>1732</td>
<td>6108</td>
<td>23248</td>
<td>90720</td>
</tr>
<tr>
<td>#multigrid iterations</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>error in $L^2$ norm</td>
<td>0.2030861</td>
<td>0.0630218</td>
<td>0.0167377</td>
<td>0.0064813</td>
<td>0.0040458</td>
</tr>
</tbody>
</table>

### Table C.10: Space Only AMR for the second 2D test at time $t = 2$

<table>
<thead>
<tr>
<th>#refinement levels</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>finest cell size</td>
<td>4/64</td>
<td>4/128</td>
<td>4/256</td>
<td>4/512</td>
<td>4/1024</td>
</tr>
<tr>
<td>#finest cells</td>
<td>872</td>
<td>2884</td>
<td>9996</td>
<td>35808</td>
<td>137344</td>
</tr>
<tr>
<td>#multigrid iterations</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>error in $L^2$ norm</td>
<td>0.4585108</td>
<td>0.1512839</td>
<td>0.0412239</td>
<td>0.0163329</td>
<td>0.0104166</td>
</tr>
</tbody>
</table>
Appendix D

Spiral Wave Results of the FitzHugh-Nagumo Model in 2D

Let us again consider the dimensionless FitzHugh-Nagumo monodomain model

\[
\frac{\partial v}{\partial t} = \epsilon \nabla \cdot \nabla v + \lambda (q - v(1 - v)(v - \theta)), \quad (D.1a)
\]

\[
\frac{\partial q}{\partial t} = \alpha v - \beta q, \quad (D.1b)
\]

for \( t > 0 \) and \( \mathbf{x} = (x_1, x_2)^T \in (0,1)^2 \), with the parameters: \( \epsilon = 0.01 \), \( \lambda = -100 \), \( \theta = 0.25 \), \( \alpha = 0.16875 \) and \( \beta = 1.0 \). The initial values are given by the smooth data

\[
v = \frac{1}{1 + e^{-50(\sqrt{x_1^2 + x_2^2} - 0.1)}} \quad \text{and} \quad q = 0,
\]

and homogeneous Neumann boundary conditions are applied. The problem is solved on the unit rectangular square \((0,1)^2\) from 0 to 12 in time with the space-time adaptive integration algorithm.

In the experiments, all of the parameters, tolerances and modules are the same as those in Example 6.5 except that a second stimulus given by

\[
I_{\text{stim}} = \begin{cases} 
1 & \text{if } \sqrt{(x_1 - 0.5)^2 + (x_2 - 0.5)^2} < 0.2 \\
0 & \text{otherwise}
\end{cases}
\]

is applied to the membrane potential around the center of the domain at time \( t = 4 \).

The adaptively refined grids and iso-contours of the membrane potential are shown in Figure D.1. The contours in each snapshot range from 0.01 to 0.99 with 10 equally distributed values.
(a) at time $t = 0$

(b) at time $t = 0.0078125$

(c) at time $t = 0.5078125$

(d) at time $t = 1.0078125$

(e) at time $t = 1.5078125$

(f) at time $t = 2.0078125$
(g) at time $t = 2.5078125$

(h) at time $t = 3.0078125$

(i) at time $t = 3.5078125$

(j) at time $t = 4.0078125$

(k) at time $t = 4.5078125$

(l) at time $t = 5.0078125$
(m) at time $t = 5.5078125$

(n) at time $t = 6.0078125$

(o) at time $t = 6.5078125$

(p) at time $t = 7.0078125$

(q) at time $t = 7.5078125$

(r) at time $t = 8.0078125$
Figure D.1: The FitzHugh-Nagumo monodomain model in two space dimensions
Appendix E

Scroll Wave Results of the FitzHugh-Nagumo Model in 3D

Let us again consider the dimensionless FitzHugh-Nagumo monodomain model

\[
\frac{\partial v}{\partial t} = \epsilon \nabla \cdot \nabla v + \lambda (q - v)(1 - v)(v - \theta), \quad (E.1a)
\]

\[
\frac{\partial q}{\partial t} = \alpha v - \beta q, \quad (E.1b)
\]

for \( t > 0 \) and \( \mathbf{x} = (x_1, x_2, x_3)^T \in (0,1)^3 \), with the parameters: \( \epsilon = 0.01, \lambda = -100, \theta = 0.25, \alpha = 0.16875 \) and \( \beta = 1.0 \). The initial values are given by the smooth data

\[
v = \frac{1}{1 + e^{-50(\sqrt{(x_1 - 0.5)^2 + (x_2 - 0.5)^2 + (x_3 - 0.5)^2 - 0.1})}} \quad \text{and} \quad q = 0,
\]

and homogeneous Neumann boundary conditions are applied. The problem is solved on the unit cube \((0,1)^3\) from 0 to 12 in time with the space-time adaptive integration algorithm.

In the experiments, all of the parameters, tolerances and modules are the same as those in Example 6.6 except that only five levels of refinement are used and a second stimulus given by

\[
I_{\text{stim}} = \begin{cases} 
1 & \text{if } \sqrt{(x_1 - 0.5)^2 + (x_2 - 0.5)^2 + (x_3 - 0.5)^2} < 0.2 \\
0 & \text{otherwise}
\end{cases}
\]

is applied to the membrane potential around the center of the domain at time \( t = 5 \).

The adaptively refined grids and iso-surfaces of the membrane potential with iso-value \( v = 0.5 \) are shown in Figure E.1.
(a) at time $t = 0$

(b) at time $t = 0.46$

(c) at time $t = 0.96$

(d) at time $t = 1.46$

(e) at time $t = 1.96$

(f) at time $t = 2.46$
(g) at time $t = 2.96$

(h) at time $t = 3.46$

(i) at time $t = 3.96$

(j) at time $t = 4.46$

(k) at time $t = 4.96$

(l) at time $t = 5.46$
(m) at time $t = 5.96$

(n) at time $t = 6.46$

(o) at time $t = 6.96$

(p) at time $t = 7.46$

(q) at time $t = 7.96$

(r) at time $t = 8.46$
Figure E.1: The FitzHugh-Nagumo monodomain model in three space dimensions
Appendix F

AMR for the Hyperbolic Conservation Laws of Gas Dynamics

As stated, the AMR algorithm has also been applied to solving hyperbolic equations by conservation laws such as the Sod’s shock-tube problem (1D), the Mach 3 wind tunnel with a step problem (2D), the oblique shock reflection problem (2D) and supersonic flow around cylinders (2D) etc.. In this appendix, we present some results of the AMR for compressible Euler equations of gas dynamics.

Figure F.1 shows the adaptively refined grids and the iso-contours of density from the simulation with AMR for the “Mach 3 wind tunnel with a step” problem (refer to the paper by Woodward and Colella [193]).

The problem begins with uniform Mach 3 flow in a wind tunnel containing a step. The wind tunnel is 1 length unit width and 3 length units along. The step is 0.2 length units high and is located 0.6 length units from the left-hand end of the tunnel.

At the left is a flow-in boundary condition, and at the right all gradients are assumed to vanish. Along the walls of the tunnel reflecting boundary conditions are applied. Initially the wind tunnel is filled with a gamma-law gas, with $\gamma = 1.4$, which everywhere has density 1.4, pressure 1.0, and velocity 3.0.

In the simulation, a total variation diminishing (TVD) scheme with a wave by wave characteristic limiting process is applied to increase numerical accuracy.

It is noticeable that the iso-contours in Figure F.1 are discontinuous across coarse-fine level interfaces since piece-wise constant cell-centered data are used in the AMR implementation for hyperbolic conservation laws.
Figure F.2 shows the adaptively refined grids and the iso-contours of density from the simulation with AMR for the double Mach reflection test problem (refer to the paper by Woodward and Colella [193] again).

This test problem describes a Mach 10 shock wave impinging on a 30 degree wedge. Transmissive/transparent boundary conditions are imposed along the boundary of the domain except that reflecting boundary conditions are applied on the wedge.

In the simulation, a total variation diminishing (TVD) scheme with the MinMod slope limiter [115] is applied to increase numerical accuracy.

In order to generate iso-contours for density, we first reconstruct continuous piece-wise linear node-centered data from piece-wise constant cell-centered output for each level grid, then a method called marching cubes is employed to extract iso-contours within each cell. The reconstructed node-centered data is still discontinuous along coarse-fine level interfaces due to the grid inconsistency there.

Both simulations use an exact Riemann Solver developed by E. F. Toro [178].
(a) at time $t = 0$

(b) at time $t = 0.25$

(c) at time $t = 0.5$
(d) at time $t = 0.75$

(e) at time $t = 1.00$

(f) at time $t = 1.25$
(g) at time $t = 1.50$

(h) at time $t = 2.00$

(i) at time $t = 2.50$
Figure F.1: The wind tunnel with a step 2D test problem for gas dynamics
Figure F.2: The oblique double Mach reflection on a 30° wedge
Bibliography


[35] Dietrich Braess and Wolfgang Hackbusch, *A new convergence proof for the


Biography

Wenjun Ying was born in the county of Yong Kang, Zhejiang Province (P. R. China) on January 29, 1976. He lived there until he matriculated at Tsinghua University (Beijing) in the fall of 1993. He was so interested in mathematics in the high school that he chose Mathematics as the major for his undergraduate studies. He received his Bachelor’s degree (B. S.) in applied mathematics from Tsinghua University in 1997 (thesis advisor: Professor Houde Han).

He came to the United States in 2000 right after his graduation from Tsinghua University with a Master’s degree (M. S.) in numerical analysis and scientific computing (thesis supervisor: Professor Houde Han). In the graduate studies at Duke University, he worked with Professor John A. Trangenstein (thesis supervisor), Professor David G. Schaeffer, Professor William K. Allard in Mathematics Department. He also worked with Professor Craig S. Henriquez and Professor Wanda Krassowska in the Department of Bio-medical Engineering (BME). He will soon start postdoctoral work at BME.

Recent Awards and Fellowships:

2004-2005 Instructorship and Fellowship from Duke University
2003-2004 Teaching Assistantship and Fellowship from Duke University
2001-2003 Research Assistantship and Fellowship from Duke University
2000-2001 Fellowship from Duke University

Awarded Visiting Activities:

2003.7-2003.7 International Workshop on Mathematical Graphics, Portland, OR
2003.1-2003.5 Visiting Student, Lawrence Livermore National Lab (LLNL), CA
2002.7-2002.8 MSRI Summer School on Blood Flow, Berkeley, CA