Hindawi Publishing Corporation
BioMed Research International
Volume 2015, Article ID 137482, 14 pages
http://dx.doi.org/10.1155/2015/137482

Research Article

Adaptive Mesh Refinement and Adaptive Time Integration for Electrical Wave Propagation on the Purkinje System

Wenjun Ying[1] and Craig S. Henriquez[2]

[1]*Department of Mathematics, MOE-LSC and Institute of Natural Sciences, Shanghai Jiao Tong University, Minhang, Shanghai 200240, China*
[2]*Departments of Biomedical Engineering and Computer Science, Duke University, Durham, NC 27708-0281, USA*

Correspondence should be addressed to Wenjun Ying; wying@sjtu.edu.cn

Received 19 November 2014; Accepted 4 February 2015

Academic Editor: Rodrigo W. dos Santos

A both space and time adaptive algorithm is presented for simulating electrical wave propagation in the Purkinje system of the heart. The equations governing the distribution of electric potential over the system are solved in time with the method of lines. At each timestep, by an operator splitting technique, the space-dependent but linear diffusion part and the nonlinear but space-independent reactions part in the partial differential equations are integrated separately with implicit schemes, which have better stability and allow larger timesteps than explicit ones. The linear diffusion equation on each edge of the system is spatially discretized with the continuous piecewise linear finite element method. The adaptive algorithm can automatically recognize when and where the electrical wave starts to leave or enter the computational domain due to external current/voltage stimulation, self-excitation, or local change of membrane properties. Numerical examples demonstrating efficiency and accuracy of the adaptive algorithm are presented.

1. Introduction

One of the long-recognized challenges in modeling cardiac dynamics [1–4] is developing efficient and accurate algorithms that can accommodate the widely varying scales in both space and time [5]. The electrical wave fronts typically occupy only a small fraction of the domain, are very sharp (in space), and change very rapidly (in time) while the electrical potential, in the region away from the wave fronts, is spatially broad and changes more slowly. With standard numerical methods on uniform grids, very small mesh parameters and very small timesteps must be used to correctly resolve the fine details of the sharp and rapidly changing wave fronts. These discretization parameters are often chosen heuristically and are fixed throughout the simulation, even if conditions change. Adaptive mesh refinement (AMR) methods have been proposed as a solution, in which coarse grids and large timesteps are used in the area where the electrical potential is changing slowly and fine grids and small timesteps are applied only in the region where the sharp electrical waves are located and the action potential changes very rapidly. Using this approach, the numbers of grid nodes and timesteps used with the adaptive algorithm are to some extent optimized. The original AMR algorithm was first proposed by Berger and Oliger for hyperbolic equations [6] and shock hydrodynamics [7]. The methods have been applied to cardiac simulations by Cherry et al. [8, 9] and Trangenstein and Kim [10].

The Berger-Oliger AMR algorithm is a hierarchical and recursive integration method for time-dependent partial differential equations. It starts time integration on a relatively coarse grid with a large timestep. The coarse grid is locally refined further if the computed solution at part of the domain is estimated to have large errors. Better solutions are obtained by continuing time integration on the fine grid with a smaller timestep until both coarse and fine grids reach the same time, called synchronization of levels. The fine grid may be locally refined further and is dynamically changing, which leads to both space and time adaptive algorithm.

The standard implementation of Berger-Oliger's AMR algorithm uses block-structured grids and assumes that

the underlying grids are logically rectangular and can be mapped onto a single index space. While the method has been shown to provide computation and accuracy advantages, it is challenging to apply to domains with complex geometry.

In this paper, we present an AMR algorithm that can be used for unstructured grids. The method is applied in both idealized and realistic tree-like domains similar to that found in the His-Purkinje system. The algorithm is based on that proposed by Trangenstein and Kim [10] where operator splitting technique is used to separate the space-independent reactions part from the linear diffusion part during time integration. Both the reactions and the diffusion parts are integrated with an implicit scheme. This allows larger and adaptive timesteps. As the AMR algorithm naturally provides a hierarchy of multilevel grids, the linear systems resulting from space discretization of the linear diffusion on the adaptively refined grids are solved by a standard geometric multigrid solver. The results show that uniform coarse discretization can lead to conduction failure or changes in dynamics in some parts of the branching network when compared to a uniform fine grid. The results also show that AMR scheme with adaptive time integration (AMR-ATI) can yield results as accurate as the uniform fine grid but with a speedup of 15 times.

The remainder of the work is organized as follows. Section 2 describes the partial differential equations, which model electrical wave propagation on the Purkinje system. Sections 3 and 4, respectively, present the time integration and space discretization for the reaction-diffusion equations. The adaptive mesh refinement and adaptive time integration procedures are outlined in Sections 5 and 6. In Section 7, some simulation results are presented with the AMR-ATI algorithm.

2. Differential Equations

Suppose that we are given a fiber network of the Purkinje system with N_V vertices and N_E edges. See Figure 1 for an idealized branch of the Purkinje system. Let d_i be the number of edges connected to vertices V_i, for each $i = 1, 2, \ldots, N_V$. We call d_i the *degree* of the vertex V_i. A vertex V_i with $d_i = 1$ is called a *leaf-vertex*. Otherwise, it is called a *nonleaf-vertex*. Denote the jth edge E_j in the fiber network by $E_j = [x_0^{(j)}, x_1^{(j)}]$. Denote the edges that have vertex V_i as the common endpoint by $E_{i_1}, E_{i_2}, \ldots, E_{i_{d_i}}$. Denote by $x_{b_{i_1}}^{(i_1)}, x_{b_{i_2}}^{(i_2)}, \ldots, x_{b_{i_{d_i}}}^{(i_{d_i})}$ the endpoints of the adjacent edges, which overlap with vertex V_i. The subscript b_{i_r} is either 0 or 1 for $r = 1, 2, \ldots, d_i$.

On each edge E_j of the fiber network, the distribution of the electric/action potential is determined by the conservation of electric currents and could be described by a partial differential equation coupled with a set of ordinary differential equations. Consider

$$\frac{a}{2}\frac{\partial J(t,x)}{\partial x} + C_m\frac{\partial u(t,x)}{\partial t} + I_{\text{ion}}(u,\mathbf{q}) = I_{\text{stim}}(t,x) \quad \text{for } x_0^{(j)} < x < x_1^{(j)}, \tag{1}$$

$$\frac{d\mathbf{q}(t,x)}{dt} = \mathscr{M}(u,\mathbf{q}),$$

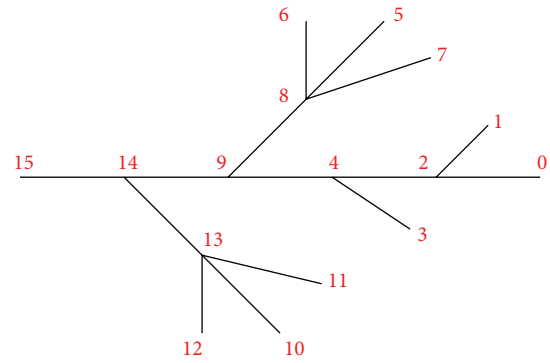


Figure 1: An idealized branch of the Purkinje system.

with

$$J(t,x) = -\frac{1}{R}\frac{\partial u(t,x)}{\partial x} \tag{2}$$

as the flux. Here, a (units: cm) denotes a typical radius of the fiber; C_m (units: μF/cm^2) is the membrane capacitance constant; R (units: kΩ·cm) is the electrical resistivity. The vector $\mathbf{q}$ denotes a vector of dimensionless gating variables. The functions $I_{\text{ion}}(u,\mathbf{q})$ and $\mathscr{M}(u,\mathbf{q})$ are typically nonlinear, describing the membrane dynamics of the fiber. One specific model is the Hodgkin-Huxley equations [11].

We assume the electric potential $u(t,x)$ is continuous,

$$u\left(t, x_{b_{i_1}}^{(i_1)}\right) = u\left(t, x_{b_{i_2}}^{(i_2)}\right) = \cdots = u\left(t, x_{b_{i_{d_i}}}^{(i_{d_i})}\right), \tag{3}$$

and the electric flux is conserved,

$$\sum_{r=1}^{d_i} (-1)^{b_{i_r}} J\left(t, x_{b_{i_r}}^{(i_r)}\right) = 0, \tag{4}$$

at each vertex V_i at any time $t > 0$. At a leaf-vertex V_i, as we have $d_i = 1$, the assumption (4) means the no-flux boundary condition is imposed.

Combined with some appropriate initial conditions for the potential function $u(t,x)$ and the gating variables $\mathbf{q}(t,x)$, the differential equations above can be uniquely solved.

3. Time Integration and Operator Splitting

We will adapt the method of lines to temporally integrate the differential equations (1). Let t^n be the discrete times, at which the equations will be discretized. At each timestep from t^n to t^{n+1}, we use an operator splitting technique to advance the space-dependent part,

$$\frac{a}{2}\frac{\partial J(t,x)}{\partial x} + C_m\frac{\partial u(t,x)}{\partial t} = 0 \quad \text{for } x_0^{(j)} < x < x_1^{(j)}, \tag{5}$$

separately from the space-independent part,

$$C_m\frac{\partial u(t,x)}{\partial t} + I_{\text{ion}}(u,\mathbf{q}) = I_{\text{stim}}(t,x), \tag{6a}$$

$$\frac{d\mathbf{q}(t,x)}{dt} = \mathscr{M}(u,\mathbf{q}), \tag{6b}$$

in the differential equations (1). Note that the space-dependent part (5) is simply a linear diffusion equation. The space-independent part (6a) and (6b) is simply a set of ordinary differential equations (ODEs).

With respect to time integration, both the linear diffusion and the nonlinear reaction parts can be in principle integrated with any standard ODE solver. In this work, we adapt implicit time integration schemes to integrate both the linear diffusion equation and the nonlinear ODEs or so-called reaction equations. An implicit scheme allows relatively larger timesteps than those imposed by the stability restriction associated with an explicit scheme.

In the next section, we will focus on the space discretization of the linear diffusion equation. For simplicity, we assume the linear diffusion equation (5) is discretized in time with the backward Euler method,

$$\frac{a}{2}\frac{\partial J\left(t^{n+1},x\right)}{\partial x}+C_m\frac{u\left(t^{n+1},x\right)-u\left(t^{n},x\right)}{\Delta t}=0 \quad \text{for } x_0^{(j)}<x<x_1^{(j)}, \tag{7}$$

with

$$J\left(t^{n+1},x\right)=-\frac{1}{R}\frac{\partial u\left(t^{n+1},x\right)}{\partial x}. \tag{8}$$

Here, $\Delta t=t^{n+1}-t^{n}$.

4. Space Discretization with the Finite Element Method

For conciseness, we will omit the time-dependency of the electric potential $u(t^{n+1},x)$ and the electric flux $J(t^{n+1},x)$. Let

$$u(x)\equiv u\left(t^{n+1},x\right),\qquad J(x)\equiv J\left(t^{n+1},x\right),\qquad b(x)\equiv \kappa u\left(t^{n},x\right) \tag{9}$$

with $\kappa=2C_m/(a\Delta t)$. Note that $b(x)$ is known in the timestep from t^n to t^{n+1}. The semidiscrete equation (7) can then be rewritten as

$$-\frac{d}{dx}\left[\frac{1}{R}u'(x)\right]+\kappa u(x)=J'(x)+\kappa u(x)=b(x) \quad \text{for } x_0^{(j)}<x<x_1^{(j)}. \tag{10}$$

In this work, we further discretize (10) with the continuous piecewise linear finite element method.

For simplicity, in this section, we only illustrate the discretization of the ODE (10) with the finite element method for the case of uniform mesh refinement.

Suppose the jth edge E_j in the fiber network is partitioned into a uniform grid, denoted by $\mathcal{G}_j$. Assume the grid $\mathcal{G}_j$ has (m_j+1) nodes, denoted by $\{\xi_i^{(j)}\}_{i=0}^{m_j}$ with $\xi_i^{(j)}=x_0^{(j)}+ih^{(j)}$ and $h^{(j)}=(x_1^{(j)}-x_0^{(j)})/m_j$.

Let $u_i^{(j)}$ be the unknown potential variable associated with the grid node $\xi_i^{(j)}$ and $\mathbf{u}_h^{(j)}=(u_0^{(j)},u_1^{(j)},\ldots,u_{m_j}^{(j)})^T$ the vector of unknown potential variables. Let

$$\varphi_i^{(j)}(\xi)=\begin{cases}\dfrac{\xi-\xi_{i-1}^{(j)}}{h^{(j)}} & \text{if } \xi_{i-1}^{(j)}<\xi<\xi_i^{(j)}\\ \dfrac{\xi_{i+1}^{(j)}-\xi}{h^{(j)}} & \text{if } \xi_i^{(j)}<\xi<\xi_{i+1}^{(j)}\\ 0 & \text{otherwise}\end{cases} \tag{11}$$

be the continuous piecewise linear finite element basis function associated with the grid node $\xi_i^{(j)}$ for each $i=1,\ldots,m_j-1$. At the endpoints $\xi_0^{(j)}=x_0^{(j)}$ and $\xi_{m_j}^{(j)}=x_1^{(j)}$, the associated basis functions read

$$\varphi_0^{(j)}(\xi)=\begin{cases}\dfrac{\xi_1^{(j)}-\xi}{h^{(j)}} & \text{if } \xi_0^{(j)}<\xi<\xi_1^{(j)}\\ 0 & \text{otherwise,}\end{cases}$$
$$\varphi_{m_j}^{(j)}(\xi)=\begin{cases}\dfrac{\xi-\xi_{m_j-1}^{(j)}}{h^{(j)}} & \text{if } \xi_{m_j-1}^{(j)}<\xi<\xi_{m_j}^{(j)}\\ 0 & \text{otherwise.}\end{cases} \tag{12}$$

Assume the finite element solution takes the form $u_h^{(j)}(x)=\sum_{i=0}^{m_j}u_i^{(j)}\varphi_i^{(j)}(x)$, which is a linear combination of the basis functions.

We introduce the electric flux $J(x)=-R^{-1}u'(x)$ at the edge endpoints $\xi_0^{(j)}=x_0^{(j)}$ and $\xi_{m_j}^{(j)}=x_1^{(j)}$ as two extra unknowns. In terms of the basis functions $\{\varphi_i^{(j)}(x)\}_{i=0}^{m_j}$, the finite element equations equivalent to the second-order ODE (10) are given by

$$\int_0^1\left[\frac{1}{R}\frac{d}{dx}u_h^{(j)}(x)\frac{d}{dx}\varphi_i^{(j)}(x)+\kappa u_h^{(j)}(x)\varphi_i^{(j)}(x)\right]dx+J\left(x_1^{(j)}\right)\varphi_i\left(x_1^{(j)}\right)-J\left(x_0^{(j)}\right)\varphi_i\left(x_0^{(j)}\right)=\int_0^1 b(x)\varphi_i^{(j)}(x)\,dx \tag{13}$$

for $i=0,1,\ldots,m_j$. At individual grid nodes, they explicitly read

$$\int_{\xi_0^{(j)}}^{\xi_1^{(j)}}\left[\frac{1}{R}\frac{d}{dx}u_h^{(j)}(x)\frac{d}{dx}\varphi_0^{(j)}(x)+\kappa u_h^{(j)}(x)\varphi_0^{(j)}(x)\right]dx-J\left(x_0^{(j)}\right)\varphi_0^{(j)}\left(x_0^{(j)}\right)=\int_{\xi_0^{(j)}}^{\xi_1^{(j)}}b(x)\varphi_0^{(j)}(x)\,dx, \tag{14}$$

$$\int_{\xi_{i-1}^{(j)}}^{\xi_{i+1}^{(j)}} \left[\frac{1}{R} \frac{d}{dx} u_h^{(j)}(x) \frac{d}{dx} \varphi_i^{(j)}(x) + \kappa u_h^{(j)}(x) \varphi_i^{(j)}(x) \right] dx = \int_{\xi_{i-1}^{(j)}}^{\xi_{i+1}^{(j)}} b(x) \varphi_i^{(j)}(x) dx \quad \text{for } i = 1, 2, \ldots, m_j - 1, \tag{15}$$

$$\int_{\xi_{m_j-1}^{(j)}}^{\xi_{m_j}^{(j)}} \left[\frac{1}{R} \frac{d}{dx} u_h^{(j)}(x) \frac{d}{dx} \varphi_{m_j}^{(j)}(x) + \kappa u_h^{(j)}(x) \varphi_{m_j}^{(j)}(x) \right] dx + J\left(x_1^{(j)}\right) \varphi_{m_j}^{(j)}\left(x_1^{(j)}\right) = \int_{\xi_{m_j-1}^{(j)}}^{\xi_{m_j}^{(j)}} b(x) \varphi_{m_j}^{(j)}(x) dx. \tag{16}$$

By further discretizing each integral in the finite element system above with the composite trapezoidal rule, we can get a set of linear equations in the following form:

$$\mathbf{A}^{(j)} \mathbf{u}_h^{(j)} + \mathbf{J}^{(j)} = h^{(j)} \mathbf{b}^{(j)}, \tag{17}$$

with $\mathbf{A}^{(j)} = (a_{r,s}^{(j)})_{(m_j+1)\times(m_j+1)}$ as the finite element stiffness matrix, $\mathbf{b}^{(j)} = (b_r^j)_{m_j+1}$ as the current vector, and $\mathbf{J}^{(j)}$ as the flux vector. The stiffness matrix $\mathbf{A}^{(j)}$ is tridiagonal and symmetric positive definite. In each row, at most three entries right on the diagonal ($a_{r,r-1}^{(j)}$, $a_{r,r}^{(j)}$, and $a_{r,r+1}^{(j)}$) are nonzero. The flux vector $\mathbf{J}^{(j)}$ has the following form:

$$\mathbf{J}^{(j)} = \left(-J\left(x_0^{(j)}\right), 0, \ldots, 0, J\left(x_1^{(j)}\right)\right)^T, \tag{18}$$

where most entries are zeros except the first and the last ones.

As the fluxes $J(x_0^{(j)})$ and $J(x_1^{(j)})$ through the endpoints of each edge E_j are unknown, the tridiagonal system (17) involves two more unknowns than equations. So, for the global system to be uniquely solvable, we need totally $2N_E$ extra equations/conditions.

Fortunately, at a nonleaf-vertex V_i, which has degree $d_i > 1$, we have $(d_i - 1)$ equations by the continuity (3) of potentials plus one more equation by the conservation (4) of electric fluxes. At a leaf-vertex V_i, which has $d_i = 1$, the electric flux conservation (4) provides exactly one boundary condition. Totally we have $\sum_{i=1}^{N_V} d_i = 2N_E$ additional equations. The number of equations in the final system is thus the same as that of unknowns. Normally, the system is well-determined.

The overall system on the fiber network involves both the potentials at the grid nodes and the electric fluxes at the fiber vertices as unknowns. In practical computation, we eliminate the electric fluxes to get a linear system with the discrete potentials as the only unknowns. As a matter of fact, noting that $\varphi_0^{(j)}(x_0^{(j)}) = 1$ and $\varphi_{m_j}^{(j)}(x_1^{(j)}) = 1$, from (14) and (16), we see the electric flux $J(x_0^{(j)})$ can be simply written out in terms of $u_0^{(j)}$ and $u_1^{(j)}$ and the electric flux $J(x_1^{(j)})$ can be simply written out in terms of $u_{m_j-1}^{(j)}$ and $u_{m_j}^{(j)}$. Plugging the explicit expressions of the electric fluxes into the conservation condition (4), we get an equation involving the unknown potentials only. After eliminating the electric fluxes, we have a well-determined system, which has $N_V + \sum_{j=1}^{N_E}(m_j - 1)$ equations and unknowns. It can be easily verified that the coefficient matrix of the resulting system is a strictly diagonally dominant and symmetric positive definite M-matrix [12].

In the case when the edge E_j is partitioned into a locally refined grid or a composite grid, the ODE (10) can be similarly discretized with the continuous piecewise linear finite element method. The coefficient matrix $\mathbf{A}^{(j)}$ of the resulting system associated with each edge E_j is also a tridiagonal, strictly diagonally dominant, and symmetric positive definite M-matrix. The overall system of discrete equations on the fiber network is well-determined too. The electric fluxes at the fiber vertices can also be eliminated so that the resulting system has the discrete potentials as the only unknowns and the coefficient matrix is a diagonally dominant M-matrix.

In this work, the grids for the space discretization are generated by local and adaptive mesh refinement (see Section 5 for details). The discrete finite element equations on the fiber network are solved with a V-cycle multigrid method. Basic components of the V-cycle iteration on the fiber network, including presmoothing, residual restriction, coarse grid correction, correction prolongation, and postsmoothing, are essentially the same as those of the V-cycle iteration for the composite grid equations on a single interval. We refer the readers to Ying's thesis work [13] for details of the multilevel/multigrid iteration.

5. Adaptive Mesh Refinement

The adaptive mesh refinement (AMR) algorithm applied for this study discretizes the fiber network into a hierarchy of dynamically, locally, and adaptively refined grids. See Figure 2 for a few composite grids, each of which consists of five locally refined grids, at different times.

The AMR algorithm follows Berger-Oliger's approach in timestepping [6–10]. It uses a multilevel approach to recursively integrate the reaction-diffusion system on the composite grids. It first integrates the system with a large timestep on a coarse level grid. Next a locally refined fine grid is generated based on available coarse data. Then the algorithm integrates the system on the fine level grid with a small timestep. Error estimation is achieved by the Richardson extrapolation [14]. By the recursive nature of the algorithm, fine grids are dynamically and locally created based on the data on coarse grids. On fine grids, smaller timesteps are used for time integration. The AMR algorithm synchronizes adjacent coarse and fine levels from time to time. At the moment of synchronization, typical routines such as mesh regridding and data up-/downscaling are performed.

Let K be the maximum number of mesh refinement levels and let ℓ_k with $k \in \{1, 2, \ldots, K\}$ be the kth level of mesh refinement. Algorithm 1 below gives a brief description of the central part of the adaptive mesh refinement procedure, which recursively advances the data at the current level ℓ_k and its finer levels.

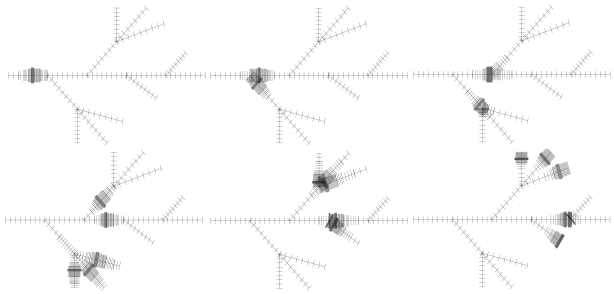


Figure 2: Adaptively refined grids from a simulation, which use five refinement levels.

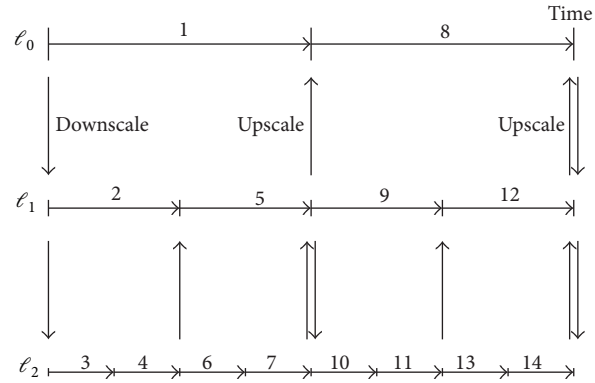


Figure 3: Recursive advancing of three mesh refinement levels.

Algorithm 1 (advance(level ℓ_k, timestep Δt_k)).

Step 1. Integrate the differential equation at the level ℓ_k by its timestep Δt_k with the Strang operator splitting technique.

(a) Integrate the split reaction equation by a half timestep $\Delta t_k/2$.

(b) Integrate the split diffusion equation by a full timestep Δt_k.

(c) Integrate the split reaction equation by a half timestep $\Delta t_k/2$.

Step 2. If the level ℓ_k has not been refined with $k < K$ or it is time to regrid the grid on the finer level ℓ_{k+1}, refine the current level ℓ_k or regrid the finer level ℓ_{k+1} after error estimation. Scale data down from the current level ℓ_k to the finer level ℓ_{k+1}.

Step 3. Set the timestep $\Delta t_{k+1} = \Delta t_k/2$ and integrate the finer level ℓ_{k+1} by two timesteps by recursively calling the function itself "advance(level ℓ_{k+1}, timestep Δt_{k+1})."

Step 4. If the finer level ℓ_{k+1} and the current level ℓ_k are synchronized, reaching the same time, scale data up from the finer level ℓ_{k+1} to the current level ℓ_k.

Figure 3 illustrates the recursive advancing or integration of three different mesh refinement levels. Coarse levels are advanced/integrated before fine levels. Each level has its own timestep of different size: a coarser level has a larger timestep size and a finer level has a smaller timestep. The algorithm first advances level ℓ_0 by Δt_0 (indicated by "1"), next advances level ℓ_1 by $\Delta t_1 = \Delta t_0/2$ (indicated by "2"), and then advances level ℓ_2 by two steps with $\Delta t_2 = \Delta t_1/2$ (indicated by "3" and "4"). Upon the synchronization of levels ℓ_1 and ℓ_2, data on the fine level ℓ_2 are upscaled to the coarser level ℓ_1. The recursive integration continues with Steps "5," "6," "7," and so forth.

The design of the AMR algorithm used in this work was based on a few assumptions proposed by Trangenstein and Kim [10].

(i) First we assume the number of elements in the base grid, which describes the computational domain and is provided by the user and is relatively small. This will allow linear systems on the grid to be solved very quickly in the middle of a multilevel (composite grid) iteration and will also improve the performance of the adaptive algorithm. The base grid is fixed during the process of adaptive mesh refinement. Other grids on fine levels are in general dynamically and recursively generated by local or uniform refinement of those on coarse levels.

(ii) Second, we assume that the region covered by the grid on a fine level is contained in the interior of that on the coarser level unless both coarse and fine grids coincide with physical boundary of the computational domain. This assumption can prevent recursive searching for element neighbors. This assumption is called *proper level nesting*.

(iii) Third, we assume that if an element of a coarse grid is refined in any part of its physical space, it must be 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 its elements. This assumption is called *alignment of level grids*.

(iv) Fourth, after the data on a coarse level grid are advanced by some time, we assume that the data on its finer level are advanced by as several timesteps 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 timestep 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 timestep on all but the finest level. This assumption is called *synchronization of advancement*.

(v) 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 timestep. This assumption is called *fine data preference*.

(vi) Sixth, 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 elements and regrid levels every coarse timestep. So, we would rather make regridding infrequently. At a coarse level, the number of timesteps between times of regridding its finer level is called *regrid interval*. In the AMR algorithm, the regrid interval may be chosen to be an integer divisor of the refinement ratio [13], which could be any even integer number in the implementation. This assumption is called *infrequent regridding*.

(vii) Finally, there is one more assumption on the adaptive algorithm, called *finite termination*. This means that the user shall specify a maximum number of refinement levels. Once the refinement reaches the maximum level, no further mesh refinement is performed.

As determined by the nature of the Purkinje system, the grids resulting from space discretization of the computational domain are essentially unstructured. This restricts direct application of Berger-Oliger's original AMR algorithm, which requires the underlying grid to be Cartesian or logically rectangular. The AMR algorithm adapted here for the Purkinje system works with unstructured grids. It represents an unstructured grid by lists of edges and nodes. A grid node is allowed to have multiple connected edges and the degree of a node could be greater than two. The unstructured grid representation can naturally describe the Purkinje system, which primarily is a tree-like structure but has some loops inside.

Finally, it is worth mentioning that in the AMR algorithm, there is a critical parameter, called AMR tolerance and denoted by tol_{AMR}, which is used by the Richardson extrapolation process for tagging coarse grid elements. The AMR tolerance closely influences efficiency and accuracy of the algorithm. The larger the tolerance is, the more efficient the algorithm is but the less accurate the solution is. The smaller the tolerance is, the more accurate the solution is but the less efficient the algorithm is. Due to the limitation of space, the detailed explanation of the Richardson extrapolation, the AMR tolerance, and other components of the AMR algorithm, such as tagging and buffering of coarse grid elements, implementation of the V-cycle multigrid iteration on the adaptively refined grids, and the data up-/downscaling between coarse and fine grids, are all omitted. We refer interested readers to Ying's dissertation [13].

6. Adaptive Time Integration

The AMR algorithm can automatically recognize when and where the wave fronts start to leave or enter the computational domain due to external current/voltage stimulation or self-excitation. Once the algorithm finds that the wave fronts have left the computational domain and determines that there is no need to make any further mesh refinement, it will turn it off and work with adaptive time integration (ATI) only. The implementation of timestep size control for the adaptive time integration is standard. In each timestep, the ODEs/reactions are integrated with a full timestep once and independently integrated with a half timestep twice. Then the two solutions are compared and an estimation of relative numerical error is obtained by the standard Richardson extrapolation technique [13]. If the estimated (maximum) relative error, denoted by $\|E\|_{\max}$, is greater than a maximum relative tolerance $\text{rtol}_{\text{ATI}}^{(\max)}$, the timestep size was rejected and a new timestep will be estimated by

$$\Delta t^{(\text{new})} = \gamma \cdot \left(\frac{\text{rtol}_{\text{ATI}}^{(\max)}}{\|E\|_{\max}} \right)^{1/(p+1)} \cdot \Delta t^{(\text{old})}, \tag{19}$$

and the Richardson extrapolation process is repeated. Otherwise, the solution with two half timesteps is simply accepted (actually an extrapolated solution could be used for better accuracy). Here, the coefficient $\gamma = 0.8$ is called a safety factor, which ensures that the new timestep size will be strictly smaller than the old one to avoid repeated rejection of timesteps. The constant p in the exponent of (19) is the accuracy order of the overall scheme.

If the suggested timestep size is less than a threshold timestep, which usually indicates that there is an abrupt/quick change of solution states, a local change of membrane properties or arrival of an external stimulus, the adaptive mesh refinement process, is then automatically turned back on again. In the implementation, the threshold timestep is not explicitly specified by the user. It is set as the timestep that is used on the base grid.

In addition, during the ATI period, if the estimated relative error $\|E\|_{\max}$ is less than a minimum relative tolerance $\text{rtol}_{\text{ATI}}^{(\min)}$, a slightly larger timestep,

$$\Delta t^{(\text{new})} = 1.1 \Delta t^{(\text{old})}, \tag{20}$$

is suggested for integration in the next step. The adaptive time integration does not like timestep size to be significantly increased within two consecutive steps for the implicit solver to have a good initial guess. For the same reason, a maximum timestep size $\Delta t^{(\max)}$ is imposed during the adaptive time integration. This usually guarantees that the Newton solver used converges within a few iterations.

7. Results

The AMR algorithm proposed in the work was implemented in custom codes written in C++. The simulations presented in this section were all performed in double precision on a dual Xeon 3.6 GHz computer. No data was output when the programs were run for timing studies.

In all of the numerical studies, the electrical resistivity R and the membrane capacitance C_m are fixed to be constant, 0.5 kΩ·cm and 1 μF/cm^2, respectively. The membrane dynamics are described by the Beeler-Reuter model [15]. No-flux (homogeneous Neumann-type) boundary conditions were applied at the leaf-nodes of the fiber network.

During the time integration for the time-dependent PDEs, a second-order operator splitting technique, the Strang splitting, is applied in each timestep. The ODEs resulting from operator splitting are integrated with a second-order singly diagonally implicit Runge-Kutta (SDIRK2) scheme. The linear diffusion equation is temporally integrated with the Crank-Nicolson scheme and spatially discretized with the continuous piecewise linear finite element method. The finite element system on locally refined grids is solved with a V-cycle multigrid/multilevel iterative method in each timestep at each refinement level. (In the V-cycle composite grid iteration, the multigrid prolongation is done by the piecewise linear interpolation. The multigrid restriction is implemented in a way so that the restriction matrix is the transpose of the prolongation matrix. At the coarsest level, as the coefficient matrix of the discrete system is a symmetric and positive definite M-matrix, the linear equations are solved with a symmetric successive overrelaxation preconditioned conjugate gradient method.) The resulting scheme has second-order global accuracy if the tolerances in each component of the adaptive algorithm are consistently selected [13].

In the adaptive simulations, the mesh refinement ratio is fixed to be two and the critical AMR tolerance tol_{AMR} is set as 0.02. The maximum and minimum relative tolerances, which are used in the ATI period, are chosen to be $\text{rtol}_{\text{ATI}}^{(\max)} = 10^{-2}$ and $\text{rtol}_{\text{ATI}}^{(\min)} = 10^{-4}$, respectively. We restrict the timestep sizes used in the adaptive time integration so that they are not greater than one millisecond, that is, $\Delta t^{(\max)} = 1$ msec. The value of p in the exponent of (19) is equal to two as the scheme has second-order accuracy.

Example 2. Simulations on a two-dimensional branch with thickness/radius-variable and nonuniform branch segments. Voltage stimulation is applied from right.

The two-dimensional branch shown in Figure 4 has a dimension of 4 cm × 2 cm, which is bounded by the rectangular domain $[0, 4] \times [1, 3]\,\text{cm}^2$. This two-dimensional branch has variable (piecewise constant) radius. Branch segments "6" and "7" have the smallest radius, equal to 20 μm. Branch segment "1" has the largest radius, equal to 160 μm and eight times that of branch segment "6." The radius of branch segment "2" is six times that of "6." Branch segments "3" and "4" have the same radius, four times that of "6." The radius of branch segment "5" is twice that of branch segment "6."

A voltage stimulation is applied from the right end of branch segment "1" through a discontinuous initial action potential, which is given as follows:

$$V_m(x, y) = \begin{cases} 25.4\,\text{mvolts} & \text{if } x > 3.5 \\ -84.6\,\text{mvolts} & \text{otherwise.} \end{cases} \quad (21)$$

The base grid used by the simulation with adaptive mesh refinement is a coarse partition of the branch structure (Figure 4) and has 240 edges with minimum element size equal to 0.0078 cm and maximum element size equal to 0.078 cm. Note that this is a highly nonuniform grid as the ratio of the maximum to the minimum element size is much greater than one. Actually, in the simulation, the base grid was generated by three times uniform bisection from a much coarser grid, which only has 30 edges. This makes the multigrid iterations for linear systems more efficient even though its contribution to the speedup of the overall algorithm is marginal as most (>90%) of the computer time used by the simulation was spent integrating nonlinear ODEs/reactions. The adaptive simulation effectively uses three refinement levels. To apply the Richardson extrapolation technique, the second level grid was created from uniform bisection of the base grid. In fact, only the grids at the third level were dynamically changing. They are regridded every other timestep.

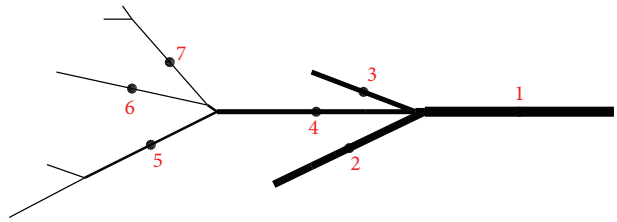


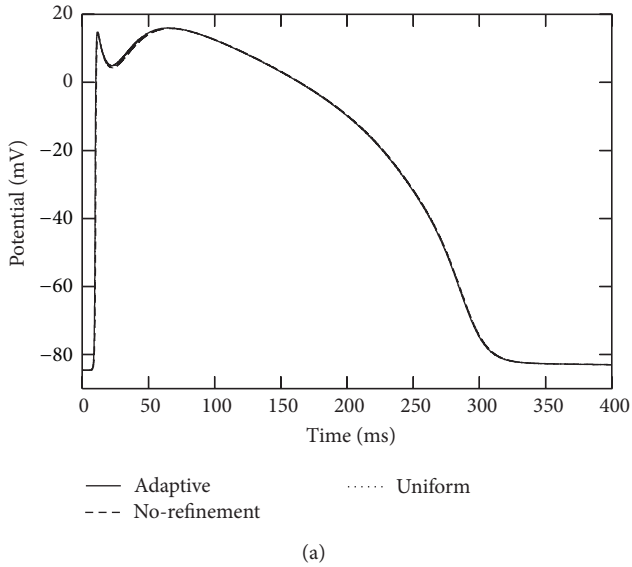
Figure 4: A typical branch in the heart conduction system is highly nonuniform in the sense that the lengths of branch segments, each of which is bounded by two adjacent leaves or branching vertices, may vary significantly. In this figure, branch segments "1" and "4" are not directly connected. Instead, they are connected through a very short branch segment. Similarly, branch segments "4" and "6" are also connected through a very short segment.

In the adaptive simulation, the timestep sizes vary from $\Delta t = 0.0098$ msecs to $\Delta t = 1.0$ msec. The AMR process was automatically turned off at time $t = 42.85$ msecs as the algorithm determines there is no need to make spatially local refinement. After that, the ATI process is activated. The simulation on the time interval $[0, 400]$ msecs totally used 99.1 secs in computer time.

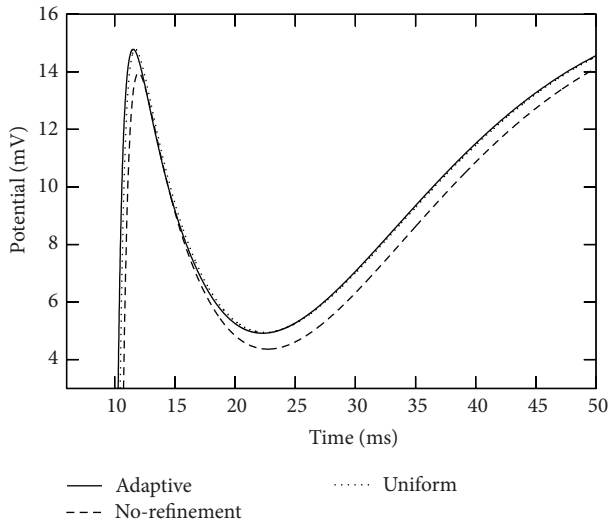
The two-dimensional branch is also partitioned into a fine, quasi-uniform grid, which has 726 edges with minimum element size equal to 0.0104 cm and maximum element size equal to 0.0127 cm. The simulation with this fine, quasi-uniform grid and timestep $\Delta t = 0.00635$ msecs on the same time interval as the adaptive one used about 1406.0 secs in computer time. We call this one "uniform" simulation.

Another simulation simply working with the coarse grid, which is used as the base grid for the AMR one, is also performed. We call this as "no-refinement" simulation. The timestep is fixed to be $\Delta t = 0.039$ msecs. This one used 76.7 secs in computer time.

In each of the simulations above, the action potential is initiated at the right end of branch segment "1" and propagates across the whole computational domain. We collected action potentials at those seven marked points (see Figure 4) during the simulations. See Figures 5 and 6 for traces of action potentials at the marked points "4" and "7," respectively. The action potentials from the adaptive and uniform mesh refinement simulations match very well, and their difference at the peak of the upstroke phase is uniformly bounded by 0.1 mV. The action potential from the "no-refinement" simulation has least accurate results. Potential

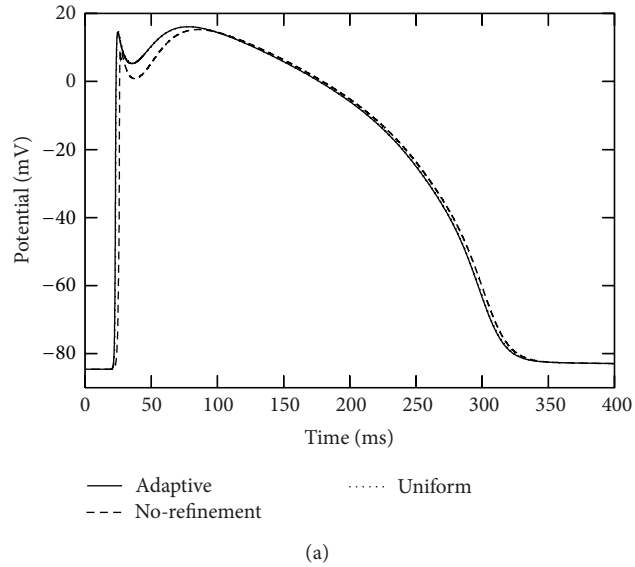


(a)

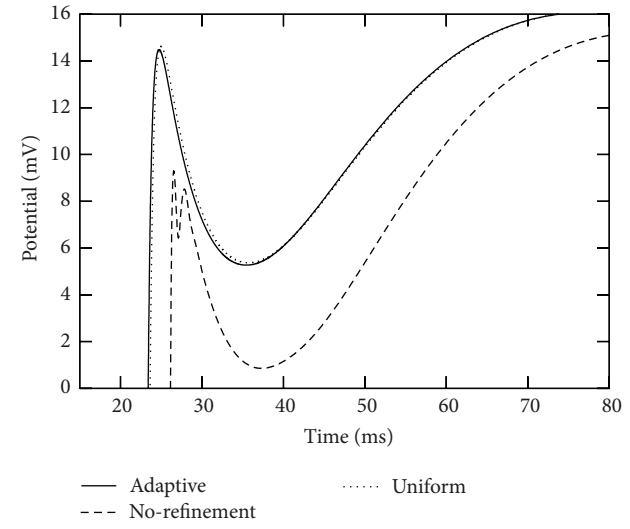


(b)

Figure 5: Traces of action potentials during the simulation period $[0, 400]$ msecs at marked point "4" in the two-dimensional branch shown in Figure 4. The solid curves were from the adaptive simulation. The dotted curves were from the uniform simulation. The dashed curves were from the "no-refinement" simulation. In these simulations, a voltage stimulation is applied from the right side of the radius-variable branch structure. The right plot is a close-up of the left plot.



(a)



(b)

Figure 6: Traces of action potentials during the simulation period $[0, 400]$ msecs at the point marked as "7" in the two-dimensional branch shown in Figure 4. The solid curves were from the adaptive simulation. The dotted curves were from the uniform simulation. The dashed curves were from the "no-refinement" simulation. In these simulations, a voltage stimulation is applied from the right side of the thickness/radius-variable branch structure. The right plot is a close-up of the left plot.

oscillation is even observed at point "7" (see Figure 6) in the "no-refinement" simulation.

Example 3. Simulations on a two-dimensional branch whose segments have piecewise constant radii. Voltage stimulation is applied from top.

We once again run three simulations, respectively, with adaptive, uniform mesh refinement and no-refinement. The computational domain and parameters are all the same as those used previously. The only difference now is to stimulate the tissue from top instead of from right.

The voltage stimulation is applied from the upper part of branch "7" through a discontinuous initial action potential, which is given as follows:

$$V_m(x, y) = \begin{cases} 25.4\,\text{mvolts} & \text{if } y > 2.8 \\ -84.6\,\text{mvolts} & \text{otherwise.} \end{cases} \quad (22)$$

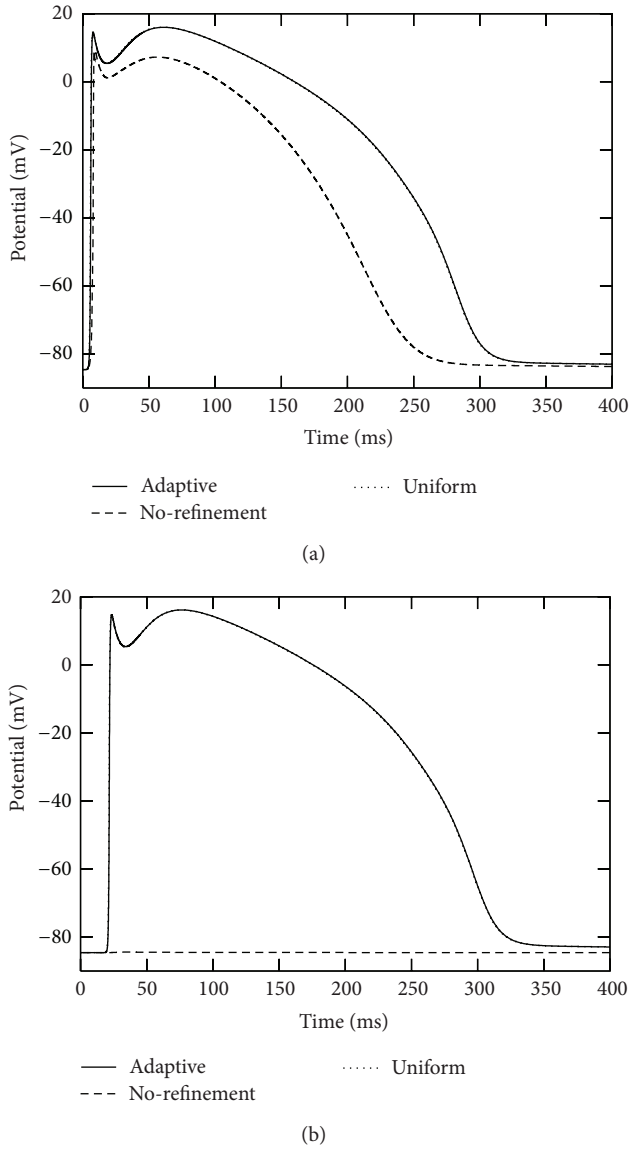


Figure 7: Traces of action potentials during the simulation period [0, 400] msecs at marked points "7" and "6" in the two-dimensional branch shown in Figure 4. The solid curves were from the adaptive simulation. The dotted curves were from the uniform simulation. The dashed curves were from the "no-refinement" simulation. In these simulations, a voltage stimulation is applied from the top of the radius-variable branch structure. The left plot corresponds to the marked point "7" and the right plot corresponds to the marked point "6."

The action potential successfully propagates across the whole computational domain in each of the adaptive and uniform refinement simulations while it fails to pass across branching points where the fiber radius changes from small to large in the "no-refinement" simulation (see Figure 7).

To verify accuracy of the solution from the adaptive simulation, action potentials at the seven marked points (see Figure 4) are also collected and compared with those from the simulation with the fine and quasi-uniform grid. It is observed that the adaptive and uniform results match very well too, and their difference at the peak of the upstroke phase is bounded by 0.15 mvolts.

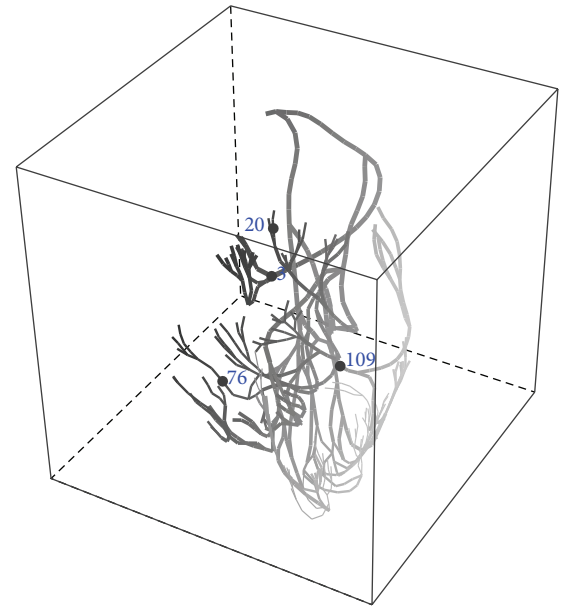


Figure 8: The idealized Purkinje system of the heart.

The simulation with adaptive mesh refinement used 100.4 msecs in computer time. The timestep sizes used in the adaptive simulation vary from $\Delta t = 0.0098$ msecs to $\Delta t = 1.0$ msec. The adaptive mesh refinement was automatically turned off at time $t = 44.02$ msecs.

The one with the uniform grid and timestep $\Delta t = 0.00635$ msecs used 1405.0 msecs in computer time.

Example 4. Simulations on a three-dimensional branch whose segments have piecewise constant radii.

The three-dimensional Purkinje system used in the following simulations was originally from 3Dscience.com. The fiber radius of the system was modified to be piecewise constant for simplicity. The radius of each branch segment takes one of the five values as the two-dimensional case. The minimum and the maximum radius of the fiber are, respectively, 20 μm and 160 μm. Intermediate values are 40, 80, and 120 cm. See Figure 8 for an illustration of the topological structure, where the ratios of variable radii are not courteously shown.

The idealized Purkinje system has a dimension $2.04 \times 2.16 \times 2.98\,\text{cm}^3$. It is bounded by the rectangular domain, $[-0.39, 1.65] \times [4.05, 6.21] \times [-1.31, 1.67]\,\text{cm}^3$. The maximum of the shortest paths from the top-most vertex to other leaf-vertices, computed by Dijkstra's algorithm is about 5.29 cm.

As in the two-dimensional examples, in each of the simulations with the idealized Purkinje system, a voltage stimulation is applied from the top branch through a discontinuous initial action potential, given as follows:

$$V_m(x, y, z) = \begin{cases} 25.4\,\text{mvolts} & \text{if } z > 1.372 \\ -84.6\,\text{mvolts} & \text{otherwise.} \end{cases} \quad (23)$$

The base grid used by the adaptive simulation is a coarse partition of the Purkinje system and has 1412 edges/elements with minimum element size equal to 0.006 cm and maximum element size equal to 0.075 cm. This is also a highly nonuniform grid as the ratio of the maximum to the minimum element size is up to 12.5. Once again, in this simulation, the base grid was actually generated from uniform bisection of a coarser grid, which has 706 edges/elements as this will make the multigrid iterations for linear systems more efficient. The adaptive simulation effectively uses three refinement levels. Similar to the two-dimensional case, the second level grid was created from uniform bisection of the base grid and only the grids at the third level were dynamically changing, regridded every other timestep.

The timestep sizes used in the adaptive simulation vary from 0.00935 msecs to 1.0 msec. The adaptive mesh refinement was automatically turned off at time $t = 42.74$ msecs. The simulation on the time interval $[0, 400]$ msecs totally used 626.7 secs in computer time.

The three-dimensional Purkinje system is also partitioned into a fine and quasi-uniform grid, which has 6311 edges/elements with minimum element size equal to 0.0095 cm and maximum element size equal to 0.016 cm. The simulation with this fine quasi-uniform grid and timestep $\Delta t = 0.0082$ msecs used about 9696.0 secs in computer time.

We also run a simulation with timestep $\Delta t = 0.0374$ msecs on the coarse grid that is used as the base grid for the adaptive simulation. This "no-refinement" simulation used 458.0 secs in computer time.

It is observed that, in each of the adaptive, uniform refinement and no-refinement simulations, the action potential successfully propagates and goes through the whole domain. The solution from the adaptive simulation is in very good agreement with that from the uniform simulation. We collected action potentials at sixteen points located in different regions of the domain and found that the corresponding potential traces almost overlap. Their difference at the peak of the upstroke phases is uniformly bounded by 0.4 mvolts. See Figures 9, 10, 11, and 12 for some visualized results, which correspond to action potentials at the four marked points shown in Figure 8. Figure 13 shows four plots of the action potential at different times by the AMR-ATI simulation. The adaptive simulation roughly uses the same computer time as the "no-refinement" one but yields much more accurate results and gains about 15.5 times speedup over the uniform one.

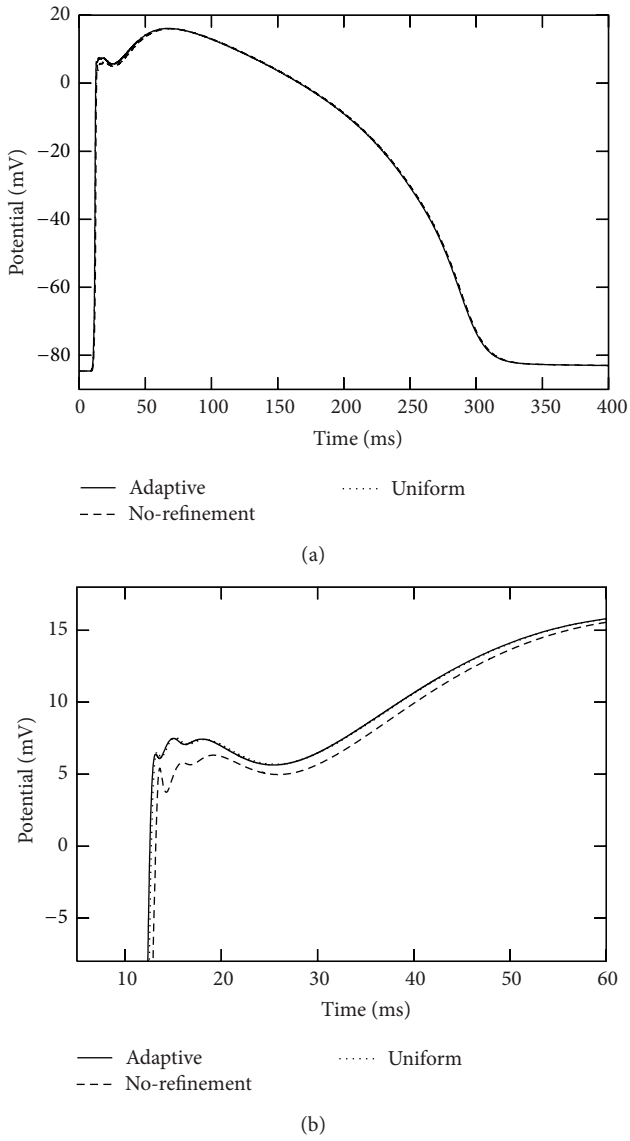


Figure 9: Traces of action potentials during the simulation period $[0, 400]$ msecs at the point marked as "3" in Figure 8. The right plot is a close-up of the left one.

8. Discussion

This paper demonstrates the promising application of a both space and time adaptive algorithm for simulating wave propagation on the His-Purkinje system.

In this work, only the numerical results with membrane dynamics described by Beeler-Reuter model are presented. The adaptive algorithm, however, is by no means restricted to the simple model. In fact, in our implementation, the algorithm successfully works with other physically realistic models, such as Luo-Rudy dynamic model and DiFrancesco-Noble model. As the advantage of the algorithm due to application of the operator splitting technique, principally any standard cardiac model of reaction-diffusion type for modeling wave propagation can be easily incorporated into the adaptive algorithm.

In addition, the adaptive algorithm can also be straightforwardly applied to modeling signal propagation on the neural system of the body or even the brain.

Other than its advantages, admittedly the algorithm has its own limitations. For example, efficiency of the adaptive algorithm heavily relies on the existence of a relatively coarse base grid, which describes the computational domain. As the speedup of an adaptive mesh refinement algorithm is roughly

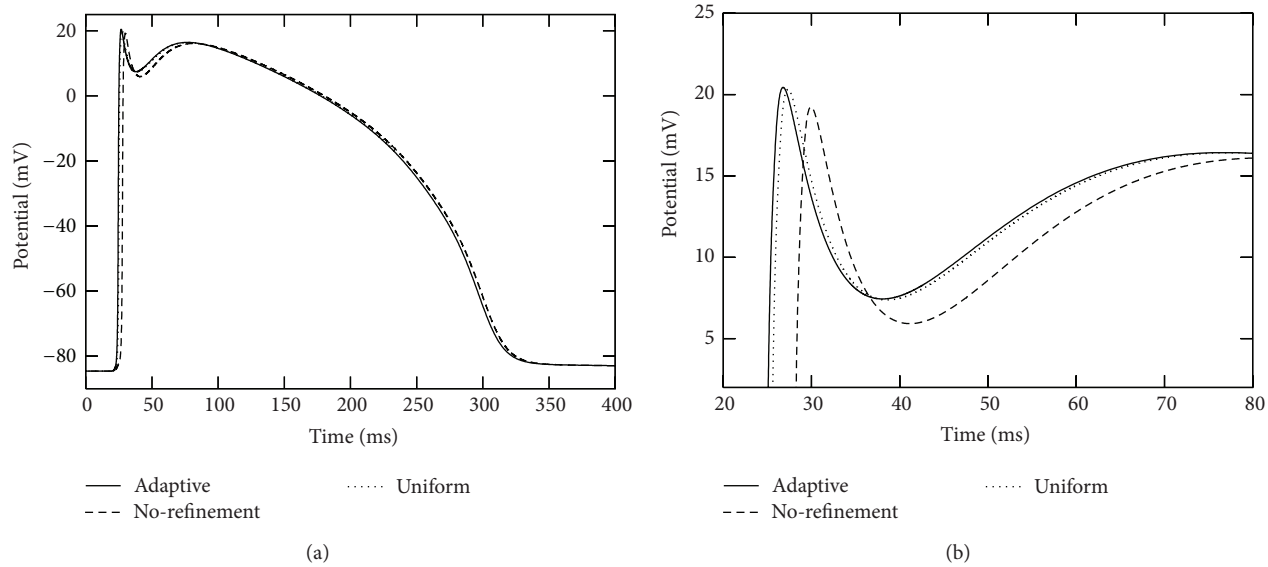


Figure 10: Traces of action potentials during the simulation period [0, 400] msecs at the point marked as "20" in Figure 8. The right plot is a close-up of the left one.

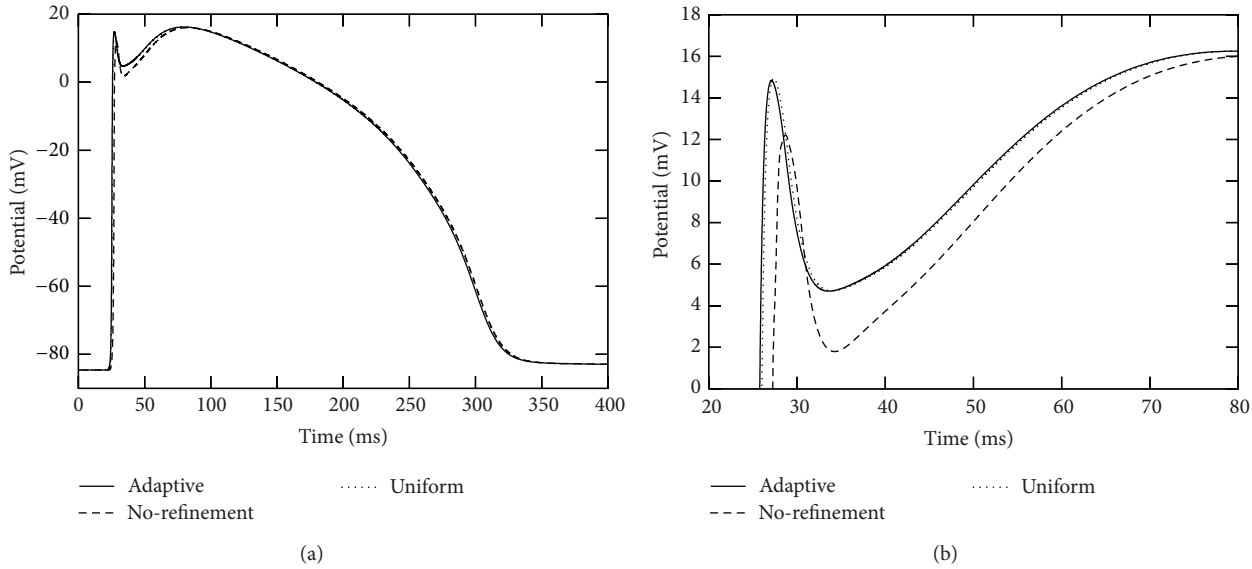


Figure 11: Traces of action potentials during the simulation period [0, 400] msecs at the point marked as "76" in Figure 8. The right plot is a close-up of the left one.

bounded by the ratio of the maximum degree of freedoms (DOFs) of the grid with the highest resolution to the mean DOFs of all grids that have ever been created during the adaptive process, the existence of a coarse base grid makes the ratio as large as possible and hence its speedup over a simulation on the grid with highest resolution. The use of a relatively coarse base grid also makes the multigrid/multilevel solver for linear systems more efficient. It is noticeable that in our simulations the base grids were all selected to have the number of edges/elements as small as possible.

In the AMR algorithm [10, 13] designed for uniform or quasi-uniform grids, the timestep size is typically selected to be proportional to the minimum of element sizes (multiplied by an estimated wave speed) in a grid in order that the numerical waves or discontinuities will never propagate more than one element within a single timestep. As this strategy guarantees that the fronts of traveling waves are always bounded and tracked by fine grids, the ODEs resulting from operator splitting applied to the reaction-diffusion systems are only integrated on the fine grids while the solution in

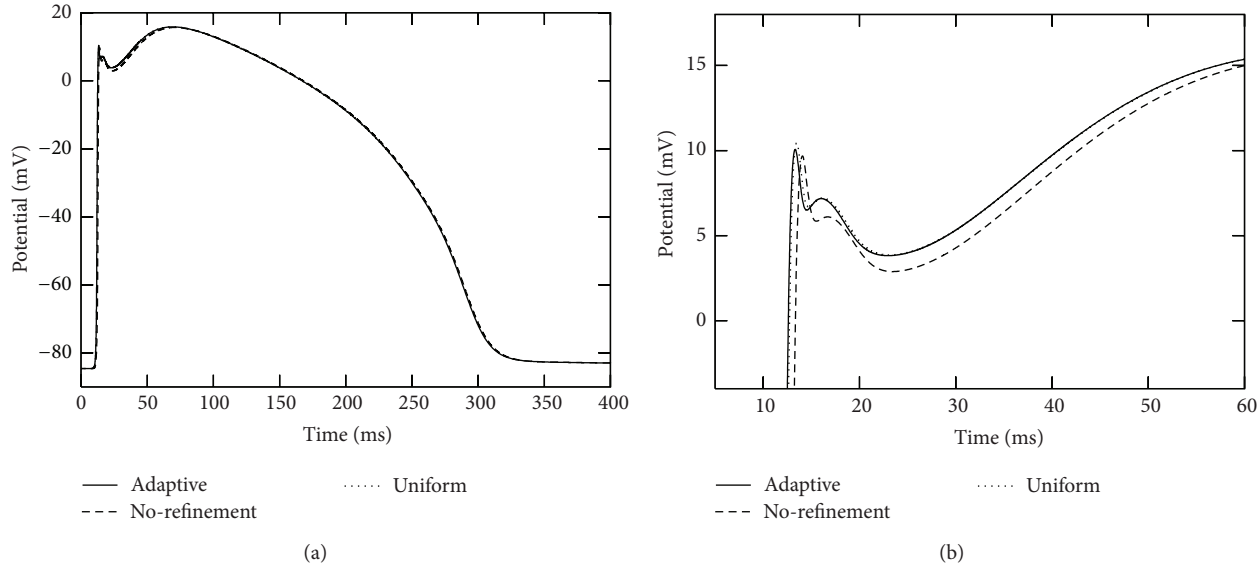


FIGURE 12: Traces of action potentials during the simulation period [0, 400] msecs at the point marked as "109" in Figure 8. The right plot is a close-up of the left one.

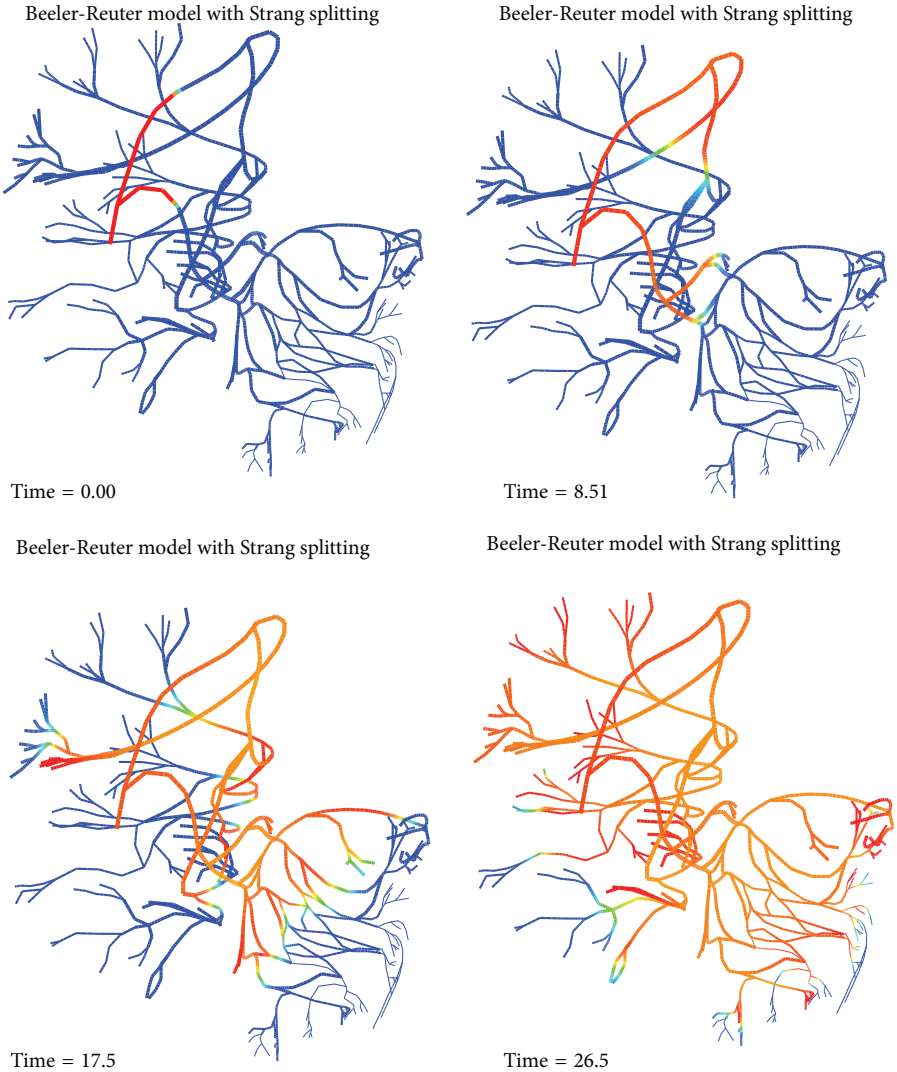


FIGURE 13: Four plots of the action potential at different times by the AMR-ATI simulation (red denotes activated potential and blue denotes inactivated potential).

other regions covered by coarser level grids is simply obtained by time interpolation.

However, a typical branch in the heart conduction system or the whole system on its own is highly nonuniform in the sense that the lengths of branch segments, each of which is bounded by two adjacent leaves or branching vertices, may vary significantly (see Figure 4), which determines that the coarse partition of the domain is also highly nonuniform since the algorithm requires the number of edges/elements as small as possible for the efficiency considerations. On this kind of computational domain, the determination of timesteps based on minimum element sizes is found to be less efficient. Here, in the proposed adaptive algorithm, timestep sizes are instead chosen to be proportional to the maximum of the element sizes in a grid. This alternative strategy is experimentally proved to be efficient and accurate enough in simulating electric waves that travel moderately fast even though its rigidity needs further investigation, in particular when the fiber radius is much larger than those currently used.

In the simulations presented in this work, error estimations in both the AMR and ATI processes were all based on the Richardson extrapolation process, which is in general more expensive but more rigorous and reliable than other simpler ones such as the gradient detector. If the gradient detector is used, the second coarsest grid in the AMR process does not need to be uniformly bisected and can also be changing dynamically, and the ATI period can use less computer time. The potential gain in algorithm efficiency and the possible loss in solution accuracy with the alternative use of a gradient detector can be studied and compared in a future work.

Finally, we make a remark about the parallelization of the space-time adaptive mesh refinement and adaptive time integration algorithm on the Purkinje system. Our numerical experiments in this work indicate that a large fraction (about 80%) of the computer time in the simulation is spent in the time integration for the reaction part while the V-cycle multigrid iteration as well as the mesh refinement part uses only a small fraction (less than 20%). The dominance of the computer time in the split reaction part is partially due to the one-dimensional nature of the Purkinje network system (mesh refinement and grid generation on one-dimensional structures are much easier than the ones with multiple dimensions). It is obvious that this dominance will be beneficial for the parallelization of the algorithm since the split reaction part (ODEs) is spatially independent, highly parallelizable, and can even be massively parallelized. Of course, the time consumption on different parts of the operator splitting depends on the specific cardiac models. With a more complicated cardiac model, the reaction part will consume more computer time than the diffusion part and the corresponding parallel algorithm is expected to have better efficiency. Admittedly, with the reaction part fully parallelized, further improvement in parallel efficiency will depend on the parallelization of the V-cycle multigrid iteration, which solves linear systems over the whole fiber network and needs time in data communication between different CPUs or multicores. The parallel performance of the AMR-ATI algorithm deserves further investigation.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

Research of the first author was supported in part by the National Science Foundation of USA under Grant DMS-0915023 and is supported by the National Natural Science Foundation of China under Grants DMS-11101278 and DMS-91130012. Research of the first author is also supported by the Young Thousand Talents Program of China.

References

[1] E. J. Vigmond and C. Clements, "Construction of a computer model to investigate sawtooth effects in the Purkinje system," *IEEE Transactions on Biomedical Engineering*, vol. 54, no. 3, pp. 389–399, 2007.

[2] E. J. Vigmond, R. W. dos Santos, A. J. Prassl, M. Deo, and G. Plank, "Solvers for the cardiac bidomain equations," *Progress in Biophysics and Molecular Biology*, vol. 96, no. 1–3, pp. 3–18, 2008.

[3] S. Linge, J. Sundnes, M. Hanslien, G. T. Lines, and A. Tveito, "Numerical solution of the bidomain equations," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 367, no. 1895, pp. 1931–1950, 2009.

[4] P. Pathmanathan, M. O. Bernabeu, R. Bordas et al., "A numerical guide to the solution of the bidomain equations of cardiac electrophysiology," *Progress in Biophysics and Molecular Biology*, vol. 102, no. 2-3, pp. 136–155, 2010.

[5] C. S. Henriquez, "A brief history of tissue models for cardiac electrophysiology," *IEEE Transactions on Biomedical Engineering*, vol. 61, no. 5, pp. 1457–1465, 2014.

[6] M. J. Berger and J. Oliger, "Adaptive mesh refinement for hyperbolic partial differential equations," *Journal of Computational Physics*, vol. 53, no. 3, pp. 484–512, 1984.

[7] M. J. Berger and P. Colella, "Local adaptive mesh refinement for shock hydrodynamics," *Journal of Computational Physics*, vol. 82, no. 1, pp. 64–84, 1989.

[8] E. M. Cherry, H. S. Greenside, and C. S. Henriquez, "A space-time adaptive method for simulating complex cardiac dynamics," *Physical Review Letters*, vol. 84, no. 6, article 1343, 2000.

[9] E. M. Cherry, H. S. Greenside, and C. S. Henriquez, "Efficient simulation of three-dimensional anisotropic cardiac tissue using an adaptive mesh refinement method," *Chaos*, vol. 13, no. 3, pp. 853–865, 2003.

[10] J. A. Trangenstein and C. Kim, "Operator splitting and adaptive mesh refinement for the Luo-Rudy I model," *Journal of Computational Physics*, vol. 196, no. 2, pp. 645–679, 2004.

[11] A. L. Hodgkin and A. F. Huxley, "A quantitative description of membrane current and its application to conduction and excitation in nerve.," *The Journal of Physiology*, vol. 117, no. 4, pp. 500–544, 1952.

[12] R. S. Varga, *Matrix Iterative Analysis*, Springer Series in Computational Mathematics, Springer, Berlin, Germany, 2nd edition, 2000.

[13] W. Ying, Department of Mathematics, Duke University, Durham, NC, USA, 2005.

[14] C. Brezinski and M. R. Zaglia, *Extrapolation Methods. Theory and Practice*, vol. 2 of *Studies in Computational Mathematics*, North-Holland Publishing, Amsterdam, The Netherlands, 1991.

[15] G. W. Beeler and H. Reuter, "Reconstruction of the action potential of ventricular myocardial fibres," *The Journal of Physiology*, vol. 268, no. 1, pp. 177–210, 1977.