OPERATOR SPLITTING AND ADAPTIVE MESH REFINEMENT FOR THE LUO-RUDY MODEL

JOHN A. TRANGENSTEIN* AND CHISUP KIM†

Abstract. We apply second-order operator splitting to the Luo-Rudy model for electrical wave propagation in the heart. This approach allows us to use *local nonlinear* iterations for the stiff nonlinear reactions, and to solve *global linear* systems for the implicit treatment of diffusion. For computational efficiency, we use dynamically adaptive mesh refinement (AMR), involving hierarchies of unions of grid patches on distinct levels of refinement. The linear system for the discretization of the diffusion on the composite AMR grid is formulated via standard conforming finite elements on unions grid patches within a level of refinement, and aligned mortar elements along interfaces between levels of refinement. The linear systems are solved iteratively by preconditioned conjugate gradients. Our preconditioner uses multiplicative domain decomposition between levels of refinement; the smoother involves algebraic additive domain decomposition between patches within a level of refinement, and Gauss-Seidel iteration within grid patches. Numerical results are presented in 1D and 2D, including spiral waves.

Key words. reaction-diffusion, excitable media, adaptive mesh refinement, operator splitting, finite elements, multigrid, domain decomposition

1. Introduction.

1.1. Modeling Electrical Activity in the Heart. Theoretical studies of electrical activity of the heart have focused on many important issues. Examples include propagation of action potentials, [34, 62], development of spiral waves [50, 65], and effects of strong electric shocks such as those used in electrical termination of arrhythmias (defibrillation) [22, 30, 36, 48, 60, 64]. Accurate calculation of the response of the heart to electrical stimuli must take into account realistic fiber geometry, anisotropy of cardiac conductivities, detailed membrane properties, microscopic tissue structure, and the inhomogeneous nature of myocardium [41, 48, 50, 59]. A model that can account for these effects, with a proper choice of parameters, is the bidomain model [31, 61], a system of reaction-diffusion partial differential equations (PDE's) coupled with a stiff ordinary differential equation (ODE) system representing cell membrane dynamics. The bidomain model consists of the equations for the intra– and extracellular potentials, Φ_i and Φ_e, coupled through the transmembrane potential, $V_m \equiv \Phi_i - \Phi_e$:

$$\nabla \cdot (\sigma_i \nabla \Phi_i) = \beta I_m, \tag{1.1a}$$

$$\nabla \cdot (\sigma_e \nabla \Phi_e) = -\beta I_m, \tag{1.1b}$$

$$I_m = C_m \frac{\partial V_m}{\partial t} + I_{ion}, \tag{1.1c}$$

where σ_i and σ_e are intra– and extracellular conductivity tensors, β is the surface-to-volume ratio of the myocardium, C_m is the membrane capacitance per unit area, and I_m is the transmembrane current per unit area. The transmembrane current is a combination of a capacitive current $C_m \partial V_m / \partial t$ and an ionic current I_{ion} which, depending on the application, is either a linear function of V_m (passive model) or a non-linear term computed from any standard model of excitable membrane [3, 21, 37] (active model). Equations (1.1) are supplemented by boundary conditions appropriate for the problem under consideration.

1.2. Survey of Numerical Methods. The complexity of the system of differential equations in heart models such as the bidomain model makes standard computer simulations expensive. This model involves two elliptic equations (1.1a) and (1.1b) for the intra-and extra-cllular potentials, and 30 to 40 reactions for various ion concentrations and gating variables in the model. This high expense lowers the utility of such models. The objective of this paper is to increase the computational efficiency of such simulations by employing advanced numerical methods. We have chosen to study the simpler Luo-Rudy model [37], which can still produce reasonably realistic simulations. The Luo-Rudy model involves a single membrane potential, and eight reactions for the calcium concentration and seven gating variables.

* Department of Mathematics, Duke University, Durham, NC 27708-0320. (johnt@math.duke.edu). This work was supported by NSF grant DMS-98-70384.

† Department of Mathematics, Duke University, Durham, NC 27708-0320. (ckim@math.duke.edu). This work was supported by NSF grant DMS-97-09608.

The numerical methods previously used in simulations 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 [52, 62] and ordinary workstations [2, 40, 49]; they use both explicit [17, 40, 49, 52] and semi– or fully implicit time–stepping techniques [2, 35, 47, 62, 63]. 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 [52]. 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 whole region of calculations.

The methods of the second group employ spatial adaptivity, temporal adaptivity, or both in order to avoid this limitation. The intent of these algorithms is to concentrate the computational work in regions of high electrical activity. Because the positive-definite conductivity tensors σ_i and σ_e are relatively small compared to the reaction rate, these regions of high electrical activity will have a small width.

At best, 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 [46] 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 (RL phase II was used in the paper) can therefore be used for simulations. A similar approach, involving a combination of an implicit integration technique with multigrid iteration is employed in [43]. A large modular code has recently been developed to accommodate a wide variety of existing cardiac models and numerical approaches in [45]. 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). 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 and thus makes the method less than optimal for defibrillation studies even in two dimensions.

Rush-Larsen temporal adaptation technique was also considered in [44]. 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.

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 has already been shown to produce a factor of 5 reduction in 2D computational time and memory expense [16], 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 [39]. Although not yet applied directly to reaction-diffusion systems the results show promise for the the spatially adaptive approach.

1.3. Overview of the Paper. The principal numerical developments in this paper are the construction of an effective multiplicative domain decomposition algorithm for such a hierarchical grid, and the development of effective synchronization strategies for adaptation in time. The issues involve both the use of multigrid-like iteration on a nested hierarchical grid for efficient solution of the diffusion equation, and the avoidance of boundary layers at interfaces between coarse and fine grid.

In section 2 we will describe the Luo-Rudy model and our numerical implementation of its model functions. In particular, we will describe how we avoid unnecessary overflow and cancellation error as well as excessive computational cost in evaluating these model functions. Next, we will describe a second-order operator splitting of reaction and diffusion in section 3. The purpose of the operator splitting is to separate the nonlinear but local reaction computations from the linear but globally coupled diffusion computations. We will describe our techniques for integrating the reactions in section 4 and our techniques for integrating the diffusion in section 5. We will use a Stiff Diagonally-Implicit Runge-Kutta (SDIRK) scheme to integrate the reactions; this scheme is both L-stable and A-stable. We will use a piecewise linear finite element method with implicit Crank-Nicolson time integration to integrate the diffusion equation. For adaptive mesh refinement (AMR) purposes, we will also describe how we formulate our discretization of diffusion on locally refined domains in section 5.4.

These computational techniques for uniform grids will give us the basic tools for developing our adaptive mesh refinement (AMR) scheme in section 6. We describe modifications to timestep selection in section 6.1. We will describe how the AMR algorithm chooses to relocate the finer mesh in section 6.2. Techniques for communicating between scales in the mesh hierarchy are described in section 6.3; this section also discusses some techniques to avoid taking fine timesteps on coarse grid. We describe modifications to the iterative linear algebra for AMR in section 7. The linear system on the hierarchical grid involves appropriate use of mortar finite elements to determine the composite grid equations; this is described in section 5.4. The iterative scheme uses a conjugate gradient iteration, preconditioned by multiplicative domain decomposition, which is described in section 7.1. In order to take advantage of the organization of the grid hierarchy, in section 7.2 we will describe a smoother that uses additive domain decomposition. This smoother works well on an array of grid patches, and extends well to distributed computing.

Finally, we will present some numerical results in section 9.

2. Description of the Luo-Rudy Model. The Luo-Rudy model is a reaction-diffusion system of the form

$$\forall \mathbf{x} \in \Omega\ ,\ \forall t > 0,\ \frac{\partial \mathbf{u}}{\partial t} \equiv \frac{\partial}{\partial t}\begin{bmatrix} \mathbf{g} \\ [Ca]_i \\ V \end{bmatrix} = \begin{bmatrix} \mathbf{a}(V) - \big(\mathbf{a}(V)+\mathbf{b}(V)\big)\mathbf{g} \\ \rho_{[Ca]_i}(\mathbf{g},[Ca]_i,V) \\ \rho_V(\mathbf{g},[Ca]_i,V) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \nabla_x \cdot D(\mathbf{x})\nabla_x V \end{bmatrix} \equiv \mathcal{R}(u) + \mathcal{D}(u) \tag{2.1a}$$

$$\forall \mathbf{x} \in \partial\Omega\ \forall t > 0,\ \mathbf{n}\cdot\nabla_x V = 0 \tag{2.1b}$$

$$\forall \mathbf{x} \in \Omega,\ \mathbf{u}(\mathbf{x},0)\text{ given} \tag{2.1c}$$

The vector $\mathbf{u}$ of unknowns consists of the vector $\mathbf{g}$ of gating variables (dimensionless), the intracellular calcium concentration $[Ca]_i$ (in milli-moles), and the membrane potential V (in milli-volts). The vectors $\mathbf{a}(V)$ and $\mathbf{b}(V)$ are vectors of reaction rates for the gating variables, and are described in equations (2.8) through (2.13) below. The notation $(\mathbf{a}+\mathbf{b})\mathbf{g}$ represents the componentwise product of the two vectors $\mathbf{a}+\mathbf{b}$ and $\mathbf{g}$. The rate of change of intracellular calcium is

$$\rho_{[Ca]_i}(\mathbf{g},[Ca]_i,V) \equiv -10^{-4} I_{si}(\mathbf{g},[Ca]_i,V) + 0.07(10^{-4} - [Ca]_i)\ , \tag{2.2}$$

where the slow inward current I_{si} (in micro-amps / square cm) is described in equation (2.5d). The rate of change of the membrane potential is

$$\rho_V(\mathbf{g},[Ca]_i,V,t) \equiv -\frac{1}{C}\left[I_{Na}(\mathbf{g},V) + I_{si}(\mathbf{g},[Ca]_i,V) + I_K(\mathbf{g},V) + I_{K_1}(\mathbf{g},V) + I_{K_p}(V) + I_b(V) + I_{st}(t)\right]\ , \tag{2.3}$$

where I_{Na} is the fast sodium current, I_{si} is the slow inward current, I_K is the time-dependent potassium current, I_{K_1} is the time-independent potassium current, I_{K_p} is the plateau potassium current, I_b is the time-independent background current and I_{st} is the stimulus current. These currents are described in equations

(2.5), and have units of micro-amps / square cm. The membrane capacitance per area is $C = 1$ micro Farad / square cm, and the membrane diffusion coefficient $D(x)$ is the membrane conductance divided by the membrane capacitance, with units of square cm / milli-second. We have taken $D(x)$ to be identically 0.00125 cm / milli-second in this paper.

The vector of gating variables has the form

$$\mathbf{g}^\top = \begin{bmatrix} K_1 & X & h & j & m & f & d \end{bmatrix} . \tag{2.4}$$

These gating variables are dimensionless, and are needed to determine the various physiological currents:

$$I_{K_1}(\mathbf{g}, V) = 0.6047\ K_1\ (V + 87.26) \tag{2.5a}$$
$$I_K(\mathbf{g}, V) = 0.282\ X\ X_i(V)\ (V + 77.) \tag{2.5b}$$
$$I_{Na}(\mathbf{g}, V) = 23\ m^3\ h\ j\ (V - 54.4) \tag{2.5c}$$
$$I_{si}(\mathbf{g}, [Ca]_i, V) = 0.09\ d\ f\ (V - 7.7 + 13.0287\ \log([Ca]_i)) \tag{2.5d}$$
$$I_{K_p}(V) = 0.0183\ K_p(V)\ (V + 87.26) \tag{2.5e}$$
$$I_b(V) = 0.03921\ (V + 59.87) \tag{2.5f}$$

In addition, the stimulus current $I_{st}(\mathbf{x}, t)$ is some specified function of space and time. We will describe our stimulus current in section 9.

The inactivation gate X_i of the time-dependent sodium current I_K (see equation (2.5b)) and the activation gate K_p of the plateau potassium current I_{K_p} (see equation (2.5e)) are assumed to reach steady state very rapidly, so they are functions of V only:

$$X_i(V) = \begin{cases} 2.837 e_1\big(-\nu_{X_i}(V)\big) \exp\big(\nu_{X_i}(V) - \delta_{X_i}(V)\big), & \nu_{X_i}(V) \geq 0 \\ 2.837 e_1\big(\nu_{X_i}(V)\big) \exp\big(-\delta_{X_i}(V)\big), & \nu_{X_i}(V) < 0 \text{ and } V > -100. \\ 1., & V \leq -100. \end{cases} \tag{2.6a}$$
where $\nu_{X_i}(V) \equiv 0.04(V + 77.)$ and $\delta_{X_i}(V) \equiv 0.04(V + 35.)$

$$K_p(V) = \begin{cases} \exp\big(-\delta_{K_p}(V)\big)/[1 + \exp\big(-\delta_{K_p}(V)\big)], & \delta_{K_p}(V) \geq 0 \\ 1/[1 + \exp\big(\delta_{K_p}(V)\big)], & \delta_{K_p}(V) < 0 \end{cases} \tag{2.6b}$$
where $\delta_{K_p}(V) \equiv (V - 7.488)/5.98$.

In order to avoid unnecessary cancellation errors, we have used the function

$$e_1(x) \equiv \frac{e^x - 1}{x} .$$

For $\mid x \mid \leq 1/4$, we use the Padé approximation

$$e_1(x) \approx \frac{15120 + x\Big(840 + x\big(420 + x(20 + x)\big)\Big)}{15120 - x\Big(6720 - x\big(1260 - x(120 - 5x)\big)\Big)} \tag{2.7}$$

with errors close to roundoff in double precision on our machine.

Recall from equation (2.1a) that the other gating variables satisfy rate equations of the form $\frac{d\mathbf{g}}{dt} = \mathbf{a} - (\mathbf{a} + \mathbf{b})\mathbf{g}$. We use the notation

$$\mathbf{a}^\top = \begin{bmatrix} \alpha_{K1} & \alpha_X & \alpha_h & \alpha_j & \alpha_m & \alpha_f & \alpha_d \end{bmatrix} ,\ \mathbf{b}^\top = \begin{bmatrix} \beta_{K1} & \beta_X & \beta_h & \beta_j & \beta_m & \beta_f & \beta_d \end{bmatrix}$$

for the vectors of rates in these rate equations. The rates for the inactivation gate K_1 of the time-independent potassium current I_{K_1} (see equation (2.5a)) are

$$\alpha_{K1}(V) = \begin{cases} 1.02\exp(-\delta_{K1}(V))/[1+\exp(-\delta_{K1}(V))], & \delta_{K1}(V) \geq 0 \\ 1.02/[1+\exp(\delta_{K1}(V))], & \delta_{K1}(V) < 0 \end{cases} \tag{2.8a}$$

where $\delta_{K1}(V) \equiv 0.2385(V + 87.26 - 59.215)$;

$$\beta_{K_1}(V) = \begin{cases} \frac{0.49124\exp(\nu_{K_1,1}(V)-\delta_{K1}(V))+\exp(\nu_{K_1,2}(V)-\delta_{K1}(V))}{1+\exp(-\delta_{K1}(V))}, & \delta_{K_1}(V) \geq 0 \\ \frac{0.49124\exp(\nu_{K_1,1}(V))+\exp(\nu_{K_1,2}(V))}{1+\exp(\delta_{K_1}(V))}, & \delta_{K_1}(V) < 0 \end{cases} \tag{2.8b}$$

where $\nu_{K_1,1}(V) \equiv 0.08032(V + 87.26 + 5.476)$, $\nu_{K_1,2}(V) \equiv 0.06175(V + 87.26 - 594.31)$, $\delta_{K_1}(V) \equiv -0.5143(V + 87.26 + 4.753)$.

The rates for the activation gate X of the time-dependent potassium current I_K (see equation (2.5b)) are

$$\alpha_X(V) = \begin{cases} 0.0005\exp(\nu_X(V) - \delta_X(V))/[1+\exp(-\delta_X(V))], & \delta_X(V) \geq 0 \\ 0.0005\exp(\nu_X(V))/[1+\exp(\delta_X(V))], & \delta_X(V) < 0 \end{cases} \tag{2.9a}$$

where $\nu_X(V) \equiv 0.083(V + 50.)$, $\delta_X(V) \equiv 0.057(V + 50.)$;

$$\beta_X(V) = \begin{cases} 0.0013\exp(\nu_X(V) - \delta_X(V))/[1+\exp(-\delta_X(V))], & \delta_X(V) \geq 0 \\ 0.0013\exp(\nu_X(V))/[1+\exp(\delta_X(V))], & \delta_X(V) < 0 \end{cases} \tag{2.9b}$$

where $\nu_X(V) \equiv -0.06(V + 20.)$, $\delta_X(V) \equiv -0.04(V + 20.)$.

The rates for the inactivation gate h of the fast sodium current I_{Na} (see equation (2.5c)) are

$$\alpha_h(V) = 0.135\exp(-(V + 80.)/6.8) \; ; \tag{2.10a}$$

$$\beta_h(V) = \begin{cases} \phi_h[3.56\exp(0.079V) + 3.1 \times 10^5\exp(0.35V)], & V < -40 \\ (1/0.13)/[1+\exp(-(V + 10.66)/11.1)], & V \geq -4 \end{cases} \tag{2.10b}$$

where ϕ_h chosen so that β_h is continuous .

The rates for the slow inactivation gate j of the fast sodium current I_{Na} (see equation (2.5c)) are

$$\alpha_j(V) = \begin{cases} -\frac{[\omega_{j1}\exp(\nu_{j1}(V)-\delta_j(V))+\omega_{j2}\exp(\nu_{j2}(V)-\delta_j(V))](V+37.78)}{1+\exp(-\delta_j(V))}, & \delta_j(V) \geq 0 \\ -\frac{[\omega_{j1}\exp(\nu_{j1}(V))+\omega_{j2}\exp(\nu_{j2}(V))](V+37.78)}{1+\exp(\delta_j(V))}, & \delta_j(V) < 0 \end{cases} \tag{2.11a}$$

where $\omega_{j1} = 1.2714 \times 10^5$, $\omega_{j2} = 3.474 \times 10^{-5}$, $\nu_{j1}(V) = 0.2444V$, $\nu_{j2}(V) = -0.04391V$, $\delta_j(V) = 0.311(V + 79.23)$;

$$\beta_j(V) = \begin{cases} 0.3, & V > 72 \\ 0.3\exp(-2.535 \times 10^{-7}V)/[1+\exp(-0.1(V+32))], & -40 \leq V \leq 72 \\ \phi_j\exp(-0.0152V)/[1+\exp(-0.1378(V+40.14))], & V < -40 \end{cases} \tag{2.11b}$$

where ϕ_j chosen so that β_j is continuous .

The rates for the activation gate m of the fast sodium current I_{Na} (see equation (2.5c)) are

$$\alpha_m(V) = \begin{cases} 3.2/e_1(-\delta_m(V)), & \delta_m(V) \geq 0 \\ 3.2\exp(\delta_m(V))/e_1(\delta_m(V)), & \delta_m(V) < 0 \end{cases} \tag{2.12a}$$

where $\delta_m(V) = 0.1(V + 47.13)$;

$$\beta_m(V) = 0.08\exp(-V/11.0) \; . \tag{2.12b}$$

The rates for the inactivation gate f of the slow inward current I_{si} (see equation (2.5d)) are

$$\alpha_f(V) = \begin{cases} 0.12\exp(\nu_f(V) - \delta_f(V))[1+\exp(-\delta_f(V))], & \delta_f(V) \geq 0 \\ 0.12\exp(\nu_f(V))[1+\exp(\delta_f(V))], & \delta_f(V) < 0 \end{cases} \tag{2.13a}$$

where $\nu_f(V) = -0.008(V + 28.)$, $\delta_f(V) = 0.15(V + 28.)$;

$$\beta_f(V) = \begin{cases} 0.0065\exp(\nu_f(V) - \delta_f(V))[1+\exp(-\delta_f(V))], & \delta_f(V) \geq 0 \\ 0.0065\exp(\nu_f(V))[1+\exp(\delta_f(V))], & \delta_f(V) < 0 \end{cases} \tag{2.13b}$$

where $\nu_f(V) = -0.02(V + 30.)$, $\delta_f(V) = -0.2(V + 30.)$.

The rates for the activation gate d of the slow inward current I_{si} (see equation (2.5d)) are

$$\alpha_d(V) = \begin{cases} 0.095\exp(\nu_d(V) - \delta_d(V))/[1+\exp(-\delta_d(V))], & \delta_d(V) \geq 0 \\ 0.095\exp(\nu_d(V))/[1+\exp(\delta_d(V))], & \delta_d(V) < 0 \end{cases} \tag{2.14a}$$

$$\text{where } \nu_d(V) = -0.01(V-5)\ ,\ \delta_d(V) = -0.072(V-5)\ ;$$

$$\beta_d(V) = \begin{cases} 0.07\exp(\nu_d(V) - \delta_d(V))/[1+\exp(-\delta_d(V))], & \delta_d(V) \geq 0 \\ 0.07\exp(\nu_d(V))/[1+\exp(\delta_d(V))], & \delta_d(V) < 0 \end{cases} \tag{2.14b}$$

$$\text{where } \nu_d(V) = -0.017(V+44.)\ ,\ \delta_d(V) = 0.05(V+44.)\ .$$

These functions have been modified slightly from the definitions in [37] to avoid unnecessary overflow and discontinuity. We do not guard against underflow. Even with our modifications, overflow can occur for very extreme values of V. On our machine, the first double-precision overflow occurs for negative V in $\beta_m(V)$ around $V = -7807$, and for positive V in $\beta_{K1}(V)$ around $V = 8831$ milli-volts. With the original Luo-Rudy model functions, the earliest overflowed occured in $\beta_{K_1}(V)$ for $V < -1472$ or $V > 1288$. In single precision, overflow in the original Luo-Rudy $\beta_{K_1}(V)$ would occur for $V < -184$ or $V > 161$ milli-volts; this is close to the range of values that occur in our simulations.

The functional forms (2.6) and (2.8) through (2.13) are very expensive to evaluate during the numerical solution of the differential equation (2.1). We use Hermite cubic spline interpolants to these functions on 20 uniform intervals over $-100 \leq V \leq 100$. These splines are as accurate in representing the Luo-Rudy model functions as those functions were accurate in representing the physical data. Further, these splines are continuously differentiable and much less expensive to evaluate.

3. Operator Splitting. The system of equations (2.1) involves both reaction and diffusion. Our approach to integrating these equations will be to use a second-order operator splitting [38, 53, 55, 58]:

$$\text{solve} \begin{cases} \frac{\partial \mathbf{u}^{\mathcal{R},n+\frac{1}{2}}}{\partial t} = \mathcal{R}(\mathbf{u}^{\mathcal{R},n+\frac{1}{2}}), & \forall \mathbf{x} \in \Omega\ ,\ \forall t^n \leq t \leq t^n + \frac{1}{2}\triangle t^{n+\frac{1}{2}} \\ \mathbf{u}^{\mathcal{R},n+\frac{1}{2}}(\mathbf{x},t^n) = \mathbf{u}(\mathbf{x},t^n), & \forall \mathbf{x} \in \Omega \end{cases} \tag{3.1a}$$

$$\text{solve} \begin{cases} \frac{\partial \mathbf{u}^{\mathcal{D},n+1}}{\partial t} = \mathcal{D}(\mathbf{u}^{\mathcal{D},n+1}), & \forall \mathbf{x} \in \Omega\ ,\ \forall t^n \leq t \leq t^n + \triangle t^{n+\frac{1}{2}} \\ \mathbf{n} \cdot \nabla_x V^{\mathcal{D},n+1}(\mathbf{x},t) = 0, & \forall \mathbf{x} \in \partial\Omega\ ,\ \forall t^n \leq t \leq t^n + \triangle t^{n+\frac{1}{2}} \\ \mathbf{u}^{\mathcal{D},n+1}(\mathbf{x},t^n) = \mathbf{u}^{\mathcal{R},n+\frac{1}{2}}(\mathbf{x},t^n + \frac{1}{2}\triangle t^{n+\frac{1}{2}}), & \forall \mathbf{x} \in \Omega \end{cases} \tag{3.1b}$$

$$\text{solve} \begin{cases} \frac{\partial \mathbf{u}^{n+1}}{\partial t} = \mathcal{R}(\mathbf{u}^{n+1}), & \forall \mathbf{x} \in \Omega\ ,\ \forall t^n + \frac{1}{2}\triangle t^{n+\frac{1}{2}} \leq t \leq t^n + \triangle t^{n+\frac{1}{2}} \\ \mathbf{u}^{n+1}(\mathbf{x},t^n + \frac{1}{2}\triangle t^{n+\frac{1}{2}}) = \mathbf{u}^{\mathcal{D},n+1}(\mathbf{x},t^{n+1}), & \forall \mathbf{x} \in \Omega \end{cases} \tag{3.1c}$$

This splitting of the problem is second-order accurate in time. The advantage of this approach is that we can integrate the reactions using stiff solvers with *localized* time step control at individual grid nodes, and we can integrate the diffusion using fast *linear* solvers. Note that others have used first-order operator splitting for the bi-domain model [54].

4. Integration of Reactions. Recall from equations (3.1a) and (3.1c) that the operator-split reaction problems have the form

$$\frac{\partial \mathbf{u}}{\partial t} = \mathcal{R}(\mathbf{u}), \forall \mathbf{x} \in \Omega\ ,\ \forall t^n < t < t^n + \triangle t$$
$$\mathbf{u}(\mathbf{x},t^n) \text{ given } \forall \mathbf{x} \in \Omega\ .$$

In practice these ordinary differential equations are stiff, meaning that $\frac{\partial \mathcal{R}}{\partial \mathbf{u}}$ has a wide range of eigenvalues. As a result, we will use appropriate stiff solvers.

4.1. Singly Diagonally Implicit Runge-Kutta Scheme. To integrate these ordinary differential equations, we will use a second-order SDIRK scheme, described in [20]. Our basic integration scheme takes the form

$$\mathbf{k}_1 = \mathcal{R}(\mathbf{u}^n + \mathbf{k}_1 \gamma \triangle t) \tag{4.1a}$$

$$\mathbf{k}_2 = \mathcal{R}(\mathbf{u}^n + [1-2\gamma]\mathbf{k}_1 \triangle t + \gamma \mathbf{k}_2 \triangle t) \tag{4.1b}$$

$$\mathbf{u}^{n+1} = \mathbf{u}^n + (\mathbf{k}_1 + \mathbf{k}_2)\frac{\triangle t}{2} \ . \tag{4.1c}$$

Here $\gamma \equiv 1 - \sqrt{1/2}$, so that the scheme is both L-stable and A-stable. The L-stability follows easily from the fact that the SDIRK scheme applied to the linear ordinary differential equation $u' = \lambda u$ leads to the approximation

$$u^{n+1} = \frac{1 + (1-2\gamma)\lambda \triangle t}{(1 - \lambda \gamma \triangle t/2)^2} u^n \equiv R(\lambda \triangle t) u^n \ ,$$

so the multiplier $R(\lambda \triangle t)$ of u^n goes to zero as $\lambda \triangle t$ becomes large. The A-stability of the scheme is also straightforward to demonstrate. Let $z = \lambda \triangle t \in \mathbf{C}$, and notice that the linear fractional transformation $w(z) = z/(1-\gamma z)$ maps the left half-plane into the interior of the circle $\mid 2\gamma w + 1 \mid \leq 1$. If $\zeta = \cos(\psi) + \imath \sin(\psi)$ is w minus the center of this circle, then $\gamma \equiv 1 - \sqrt{1/2}$ implies that the amplification factor R satisfies

$$\mid R \mid^2 = \frac{5}{8} + \frac{1}{2}\cos\psi - \frac{1}{8}\cos 2\psi \ .$$

The extreme points of this function occur at $\psi = 0, \pi$ and 2π, where $R = 1, 0$ and 1, respectively. Thus $\mid R \mid^2 \leq 1$ for all z satisfying $\mid 2\gamma w + 1 \mid \leq 1$, or equivalently for all z in the left half-plane. Extensions of this scheme to higher order can be found in [32, 42].

Another common choice is the TRBDF scheme

$$\frac{\mathbf{u}^{n+\frac{1}{2}} - \mathbf{u}^n}{\frac{1}{2}\triangle t} = \frac{1}{2}[\mathcal{R}(\mathbf{u}^{n+\frac{1}{2}}) + \mathcal{R}(\mathbf{u}^n)] \tag{4.2a}$$

$$\frac{\frac{3}{2}\mathbf{u}^{n+1} - 2\mathbf{u}^{n+\frac{1}{2}} + \frac{1}{2}\mathbf{u}^n}{\frac{1}{2}\triangle t} = \mathcal{R}(\mathbf{u}^{n+1}) \tag{4.2b}$$

For the ordinary differential equation $u' = \lambda u$, the TRBDF scheme leads to the approximation

$$u^{n+1} = \frac{1 + \frac{5}{12}\lambda \triangle t}{(1 - \frac{1}{3}\lambda \triangle t)(1 - \frac{1}{4}\lambda \triangle t)} u^n \ .$$

As approximations to the exponential $\exp(\lambda \triangle t)$, the amplification factors for the SDIRK and TRBDF schemes are very similar. The TRBDF scheme has negative amplification factor for $\lambda \triangle t < -12/5 = -2.4$, while the SDIRK scheme has negative amplification factor for $\lambda \triangle t < -\sqrt{2} - 1 \approx -2.414$. The SDIRK scheme, however, involves the same matrix in the linear system for Newton's method in both of the nonlinear equations, while TRBDF involves different linear systems.

4.2. Solving the Nonlinear Systems. The SDIRK scheme equations (4.1a) and (4.1b) require the solution of nonlinear equations of the form

$$\mathbf{f}(\mathbf{k}) \equiv \mathbf{k} - \mathcal{R}(\mathbf{u} + \mathbf{k}\gamma \triangle t) = 0$$

for fixed $\mathbf{u}$ and fixed $\triangle t$. Newton iteration to solve such a nonlinear system replaces $\mathbf{k}$ by $\mathbf{k} + \mathbf{s}$ where $\mathbf{s}$ solves

$$\frac{\partial \mathbf{f}}{\partial \mathbf{k}} \mathbf{s} = -\mathbf{f}(\mathbf{k}) \ .$$

Note that the Jacobian matrix $\frac{\partial \mathbf{f}}{\partial \mathbf{k}}$ has a special structure for the Luo-Rudy model:

$$\frac{\partial \mathbf{f}}{\partial \mathbf{k}} = \begin{bmatrix} I + \mathbf{diag}(\mathbf{a}) + \mathbf{diag}(\mathbf{b}) & 0 & -\mathbf{a}' + (\mathbf{a}' + \mathbf{b}')\mathbf{g} \\ -\frac{\partial \rho_{[Ca]_i}}{\partial \mathbf{g}} & 1 - \frac{\partial \rho_{[Ca]_i}}{\partial [Ca]_i} & -\frac{\partial \rho_{[Ca]_i}}{\partial V} \\ -\frac{\partial \rho_V}{\partial \mathbf{g}} & -\frac{\partial \rho_V}{\partial [Ca]_i} & 1 - \frac{\partial \rho_V}{\partial V} \end{bmatrix} \equiv \begin{bmatrix} \mathbf{D} & \mathbf{G} \\ \mathbf{F} & \mathbf{T} \end{bmatrix}$$

where $\mathbf{D} = I + \mathbf{diag}(\mathbf{a}) + \mathbf{diag}(\mathbf{b})$ is 7×7 and diagonal. As a result, it is easy to use the first 7 equations to eliminate the increments for the gating variables from the system of 9 Newton equations, and reduce to a system of 2 equations involving the increments for the intracellular calcium $[Ca]_i$ and the membrane potential V. Specifically, in order to solve

$$\begin{bmatrix} \mathbf{D} & \mathbf{G} \\ \mathbf{F} & \mathbf{T} \end{bmatrix} \begin{bmatrix} \mathbf{s_g} \\ \mathbf{s}_2 \end{bmatrix} = - \begin{bmatrix} \mathbf{f_g} \\ \mathbf{f}_2 \end{bmatrix}$$

we compute

$$\begin{aligned} \mathbf{M} &= \mathbf{T} - \mathbf{F}\mathbf{D}^{-1}\mathbf{G} \in \mathbf{R}^{2\times 2} \\ \mathbf{s}_2 &= -\mathbf{M}^{-1}(\mathbf{f}_2 - \mathbf{F}\mathbf{D}^{-1}\mathbf{f_g}) \in \mathbf{R}^2 \\ \mathbf{s_g} &= -\mathbf{D}^{-1}(\mathbf{f_g} + \mathbf{G}\mathbf{s}_2) \in \mathbf{R}^7 \end{aligned}$$

Any linear systems involved in Newton's method are either 7×7 diagonal (involving $\mathbf{D}$) or 2×2 (involving $\mathbf{M}$).

4.3. Comments on Reaction Integrators. Our use of second-order integration methods for reaction is consistent with our use of second-order operator splitting of the reaction and diffusion (described in section 3), with our second-order integration of diffusion (described in section 5), and with our use of continuously differentiable cubic splines to represent the gate rates $\mathbf{a}(V)$ and $\mathbf{b}(V)$ (described in section 2). However, stimulus currents $I_{st}(\mathbf{x}, t)$ that involve discontinuities in time can prevent our reaction integrations from exhibiting second-order accuracy.

It is reasonable to ask whether higher-order integration schemes for the reactions would be appropriate. In theory, this would make sense only if we were able to produce a fully third-order algorithm for integrating the Luo-Rudy model equations. In order to produce a third-order algorithm for the Luo-Rudy model, it would be necessary to use twice continuously differentiable splines to represent $\mathbf{a}(V)$ and $\mathbf{b}(V)$ (to increase the regularity of the solution of the differential equations), to use third-order integration of diffusion and third-order operator splitting of reaction and diffusion. If the computational domain is rectangular, then it would also be necessary to perform local refinement near the corners to prevent degradation of accuracy due to restricted regularity of the solution near the corners; alternatively, we could modify the finite element method for the diffusion to include basis functions that represent the non-smooth behavior near the corners.

In practice, sometimes accuracy is a more important consideration than order. In other words, it is possible that a reaction integrator with higher than second order accuracy might obtain the desired absolute accuracy with less work than our SDIRK scheme. Such decisions depend on other factors, such as the choice of mesh width and timestep. For example, with relatively coarse meshes the propagating reaction front will be very sharp, and the reaction integration will have to deal with very large changes in the solution in one timestep. On the other hand, with relatively fine meshes the diffusion will spread the reaction front over a moderate number of grid cells, and the reaction integration will see much smaller changes in the solution in one timestep.

We have implemented a Richardson extrapolation of our SDIRK integration, in order to achieve a desired accuracy, possibly through high order. Richardson extrapolation involves recursive timestep halving, so the work goes up substantially with each extrapolation. For example, the second-order SDIRK integration requires two nonlinear system solves, so the third-order extrapolant requires an additional four nonlinear solves, fourth-order requires an additional eight nonlinear solves, and so on. Numerical experiments indicate that these extrapolants almost always achieve the desired order of accuracy; only abrupt changes in the stimulus current or occasional straddling of spline interpolation points prevents high order in extrapolation. The reason for using the Richardson extrapolation with the SDIRK scheme is to test the usefulness of higher-order reaction integration. We will discuss this issue further with our numerical results in section 9.

In order to determine an appropriate tolerance for the reaction integration, we first determined a model for the error due to operator splitting and spatial discretization of the diffusion equation. In other words, we chose very small tolerances for the reaction integration, for the nonlinear system solves and for the iterative solution of the linear systems for the diffusion equation. We performed mesh refinement studies to see how the error behaved as a function of the mesh width. This gave us the following empirical formula for the reaction tolerance:

$$\text{tolerance } = \min\{10^{-3}, \max\{10^4\epsilon, 40\,dx^2\}\}$$

where ϵ is machine roundoff. In other words, Newton iteration is continued until

$$|\ \mathbf{f}_2(\mathbf{k})\ | \leq \text{tolerance} \max\{|\ \mathbf{k}_2\ |, |\ \mathcal{R}_2\ |\}$$

and Richardson extrapolation of the SDIRK scheme is continued until

$$|\ \Delta V\ | \leq \text{tolerance } |\ V\ |$$

where ΔV is the estimated error in the Richardson extrapolation for the membrane potential V.

5. Integration of Diffusion. In order to integrate the diffusion equation (3.1b) in second-order operator splitting, we will develop a finite element discretization that is second-order accurate in both space and time. On a grid without local refinement, we will use piecewise-linear functions for spatial discretization and Crank-Nicolson for temporal discretization. This approach is described in sections 5.2 and 5.3. With local refinement refinement, we will use mortar elements to form the composite grid equations in section 5.4.

5.1. Weak Form on the Full Domain. Given initial data $V_0 \in H^0(\Omega)$, the weak form of the diffusion problem (3.1b) is to find $V(x,t)$ so that $\forall t > 0$ $V(\cdot,t) \in H^1(\Omega)$, $V(\cdot,0) \in L^2(\Omega)$ and

$$\begin{aligned} &\forall w \in H^1(\Omega), \quad 0 = \frac{d}{dt}\int_\Omega w(x)V(x,t)\ dx + \int_\Omega \nabla_x w(x)\cdot D(x)\nabla_x V(x,t)\ dx \equiv \frac{d}{dt}(w,V) + \mathcal{B}(w,V) \\ &\forall w \in L^2(\Omega), \quad 0 = (w, V - V_0)\ . \end{aligned} \tag{5.1}$$

Here $H^1(\Omega)$ represents the usual Sobolev space, consisting of functions whose derivatives of order up to one are square-integrable on Ω. We assume that in any number of dimensions, the diffusion coefficient is bounded and uniformly positive-definite:

$$\exists \overline{D} \geq \underline{D} > 0 \quad \forall x \in \Omega \quad \forall y \neq 0 \quad \underline{D} \leq \frac{y^\top D(x) y}{y^\top y} \leq \overline{D}\ .$$

As a result, the bilinear form $\mathcal{B}$ is self-adjoint, bounded and coercive on $H^1(\Omega)$. This problem is known to be equivalent to

$$\begin{aligned} \frac{\partial V}{\partial t} &= \nabla_x \cdot D(\mathbf{x})\nabla_x V \ \forall \mathbf{x} \in \Omega\ \forall t > 0 \\ \mathbf{n}\cdot\nabla_x V &= 0\ \forall \mathbf{x} \in \partial\Omega\ \forall t > 0 \\ V(\mathbf{x},t) &= V_0(\mathbf{x})\ \forall \mathbf{x} \in \Omega \end{aligned}$$

provided that the initial data V_0, the diffusion coefficient D and the domain Ω are sufficiently smooth.

5.2. Galerkin Methods on the Full Domain. To integrate the diffusion equation in the operator-splitting subproblem (3.1b) we use a conforming finite element method. An excellent reference for this technique applied to diffusion problems is [56]. We begin by choosing a finite dimensional subspace $\mathcal{M}_h \subset H^1(\Omega)$; our choice of $\mathcal{M}_h$ is described below. The semi-discrete Galerkin approximation is a function $V_h(x,t) \in \mathcal{M}_h$ for all $t > 0$ that satisfies the weak equations

$$\begin{aligned} \forall W_h \in \mathcal{M}_h,\quad 0 &= \frac{d}{dt}\int_\Omega W_h(x)V_h(x,t)\ dx + \int_\Omega \nabla_x W_h(x)\cdot D(x)\nabla_x V_h(x,t)\ dx && \text{(5.2a)} \\ &\equiv \frac{d}{dt}(W_h,V_h) + \mathcal{B}(W_h,V_h)\ , \\ \forall W_h \in \mathcal{M}_h,\quad 0 &= (W_h, V_0 - V_h)\ . && \text{(5.2b)} \end{aligned}$$

In (5.2b) we have used the L^2-projection of the initial data for the semi-discrete problem. Note that if $V_0 \in \mathcal{M}_h$, then the L^2 projection implies that $V_h(x,0) = V_0(x)$.

A fully discrete Galerkin approximation uses an appropriate stiffly stable method for solving initial value problems for ordinary differential equations. A finite element method for approximating the solution of (5.1) is a Galerkin method that chooses the finite-dimensional subspace $\mathcal{M}_h$ to consist of piecewise polynomials on some mesh of width h. In our applications, we will choose $\mathcal{M}_h$ to consist of continuous functions that are linear in each coordinate on some given rectangular mesh. We will let h denote the maximum cell width in any coordinate direction. It is well-known that for quasi-uniform meshes, $\mathcal{M}_h$ approximates functions in $H^2(\Omega)$ with order h^2. We will use the trapezoidal rule to integrate in time; this leads to the familiar Crank-Nicolson method for the diffusion equation:

$$\forall W_h \in \mathcal{M}_h, \quad 0 = \frac{1}{\triangle t}(W_h, V_h(\cdot,t^{n+1}) - V_h(\cdot,t^n)) + \frac{1}{2}\mathcal{B}(W_h, V_h(\cdot,t^{n+1}) + V_h(\cdot,t^n)) \;, \tag{5.3a}$$

$$\forall W_h \in \mathcal{M}_h, \quad 0 = (W_h, V_0 - V_h(\cdot,0)) \;. \tag{5.3b}$$

It is possible to show [56] that the error in the fully discrete method satisfies

$$\begin{aligned}\|V(\cdot,t) - V_h(\cdot,t)\|_{H^0(\Omega)} \le & C \Bigg[e^{-\lambda_1 t}\|V(\cdot,0) - V_h(\cdot,0)\|_{H^0(\Omega)} + h^2 \int_0^t \|\frac{\partial V}{\partial t}(\cdot,s)\|_{H^2(\Omega)} ds \\ & + \triangle t^2 \int_0^t \|\frac{\partial^3 V}{\partial t^3}(\cdot,s)\|_{H^0(\Omega)} ds + \triangle t^2 \int_0^t \|\frac{\partial^2 \nabla_x \cdot D \nabla_x V}{\partial t^3}(\cdot,s)\|_{H^0(\Omega)} ds \Bigg] \;.\end{aligned}$$

Here C is a constant depending on Ω and D, and λ_1 is the smallest eigenvalue of $-\nabla_x \cdot D \nabla_x$ on Ω. The scheme is basically second-order in space and time for $t = O(1)$; at early time, the order of the error depends on the accuracy with which the initial data can be approximated by the finite element space. Note that the influence of the error in the initial data fades exponentially as time increases.

It is well-known that the weak equations using Crank-Nicolson time integration lead to linear systems of the form

$$(\mathbf{M} + \mathbf{K}\frac{\triangle t^{n+\frac{1}{2}}}{2})\mathbf{V}^{n+1} = (\mathbf{M} - \mathbf{K}\frac{\triangle t^{n+\frac{1}{2}}}{2})\mathbf{V}^n \tag{5.4}$$

Here $\mathbf{M}$ is the mass matrix, and $\mathbf{K}$ is the stiffness matrix. We will develop the details of these matrices in section 5.3 below.

5.3. Numerical Implementation. In 1D, note that the nonzero contributions to the mass or stiffness matrices from element (x_k, x_{k+1}) only involve nodal basis functions W_k and W_{k+1}. For piecewise constant diffusion coefficient D, the integrals can be performed exactly; otherwise, it is necessary to sufficient to approximate the integrals using a one-point Gaussian quadrature rule (*i.e.*, the midpoint rule). This suggests that we form the diffusive quadratures

$$\begin{aligned}\begin{bmatrix} q_0 \\ q_1 \end{bmatrix}_{k+\frac{1}{2}} \equiv & \int_{x_k}^{x_{k+1}} \begin{bmatrix} W_k(x) \\ W_{k+1}(x) \end{bmatrix} \begin{bmatrix} W_k(x) & W_{k+1}(x) \end{bmatrix} \, dx \begin{bmatrix} v_k^{n+1} - v_k^n \\ v_{k+1}^{n+1} - v_{k+1}^n \end{bmatrix} \\ & + \triangle t^{n+\frac{1}{2}} \int_{x_k}^{x_{k+1}} \begin{bmatrix} \frac{\partial W_k}{\partial x} \\ \frac{\partial W_{k+1}}{\partial x} \end{bmatrix} D(x) \begin{bmatrix} \frac{\partial W_k}{\partial x} & \frac{\partial W_{k+1}}{\partial x} \end{bmatrix} \, dx \begin{bmatrix} \frac{v_k^{n+1} + v_k^n}{2} \\ \frac{v_{k+1}^{n+1} + v_{k+1}^n}{2} \end{bmatrix} \\ = & \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} v_k^{n+1} - v_k^n \\ v_{k+1}^{n+1} - v_{k+1}^n \end{bmatrix} \frac{dx_{k+\frac{1}{2}}}{6} + \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} v_k^{n+1} + v_k^n \\ v_{k+1}^{n+1} + v_{k+1}^n \end{bmatrix} \frac{D_{k+\frac{1}{2}} \triangle t^{n+\frac{1}{2}}}{2\, dx_{k+\frac{1}{2}}} \\ \equiv & \mathbf{M}_{k+\frac{1}{2}} \begin{bmatrix} v_k^{n+1} - v_k^n \\ v_{k+1}^{n+1} - v_{k+1}^n \end{bmatrix} + \mathbf{K}_{k+\frac{1}{2}} \frac{\triangle t^{n+\frac{1}{2}}}{2} \begin{bmatrix} v_k^{n+1} + v_k^n \\ v_{k+1}^{n+1} + v_{k+1}^n \end{bmatrix}\end{aligned} \tag{5.5}$$

The finite element equations at internal nodes take the form

$$(q_0)_{k+\frac{1}{2}} + (q_1)_{k-\frac{1}{2}} = 0 \tag{5.6}$$

In general, the sum, of the quadratures associated with a given node, over elements within the computational domain is zero. The mass matrix $\mathbf{M}$ and stiffness matrix $\mathbf{K}$ in equation (5.4) are formed by combining the elementwise mass and stiffness matrices from equations (5.5) into the finite element equations (5.6).

In 2D, the nonzero contributions to the mass or stiffness matrices from element $(x_k, x_{k+1}) \times (y_\ell, y_{\ell+1})$ involve only nodal basis functions $W_{k,\ell}$, $W_{k+1,\ell}$, $W_{k,\ell+1}$ and $W_{k+1,\ell+1}$. For piecewise constant diffusion coefficient D, the integrals can be performed exactly. This suggests that we form the diffusive quadratures

$$
\begin{aligned}
&\begin{bmatrix} q_{00} \\ q_{10} \\ q_{01} \\ q_{11} \end{bmatrix}_{k+\frac{1}{2},\ell+\frac{1}{2}} \qquad (5.7) \\
&\equiv \int_{y_\ell}^{y_{\ell+1}} \int_{x_k}^{x_{k+1}} \begin{bmatrix} W_{k,\ell} \\ W_{k+1,\ell} \\ W_{k,\ell+1} \\ W_{k+1,\ell+1} \end{bmatrix} \begin{bmatrix} W_{k,\ell} & W_{k+1,\ell} & W_{k,\ell+1} & W_{k+1,\ell+1} \end{bmatrix} dx \begin{bmatrix} \upsilon^{n+1}_{k,\ell} - \upsilon^n_{k,\ell} \\ \upsilon^{n+1}_{k+1,\ell} - \upsilon^n_{k+1,\ell} \\ \upsilon^{n+1}_{k,\ell+1} - \upsilon^n_{k,\ell+1} \\ \upsilon^{n+1}_{k+1,\ell+1} - \upsilon^n_{k+1,\ell+1} \end{bmatrix} \\
&+ \frac{\triangle t^{n+\frac{1}{2}}}{2} \int_{y_\ell}^{y_{\ell+1}} \int_{x_k}^{x_{k+1}} \begin{bmatrix} (\nabla_x W_{k,\ell})^\top \\ (\nabla_x W_{k+1,\ell})^\top \\ (\nabla_x W_{k,\ell+1})^\top \\ (\nabla_x W_{k+1,\ell+1})^\top \end{bmatrix} D \begin{bmatrix} \nabla_x W_{k,\ell} & \nabla_x W_{k+1,\ell} & \nabla_x W_{k,\ell+1} & \nabla_x W_{k+1,\ell+1} \end{bmatrix} dx\, dy \\
&\quad \begin{bmatrix} \upsilon^{n+1}_{k,\ell} + \upsilon^n_{k,\ell} \\ \upsilon^{n+1}_{k+1,\ell} + \upsilon^n_{k+1,\ell} \\ \upsilon^{n+1}_{k,\ell+1} + \upsilon^n_{k,\ell+1} \\ \upsilon^{n+1}_{k+1,\ell+1} + \upsilon^n_{k+1,\ell+1} \end{bmatrix} \\
&= \begin{bmatrix} 4 & 2 & 2 & 1 \\ 2 & 4 & 2 & 2 \\ 2 & 2 & 4 & 2 \\ 1 & 2 & 2 & 4 \end{bmatrix} \begin{bmatrix} \upsilon^{n+1}_{k,\ell} - \upsilon^n_{k,\ell} \\ \upsilon^{n+1}_{k+1,\ell} - \upsilon^n_{k+1,\ell} \\ \upsilon^{n+1}_{k,\ell+1} - \upsilon^n_{k,\ell+1} \\ \upsilon^{n+1}_{k+1,\ell+1} - \upsilon^n_{k+1,\ell+1} \end{bmatrix} \frac{dx_{k+\frac{1}{2}}\, dy_{\ell+\frac{1}{2}}}{36} \\
&+ \frac{\triangle t^{n+\frac{1}{2}}}{2} \left\{ \begin{bmatrix} -1 & -1 \\ 1 & -1 \\ -1 & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} dx_{k+\frac{1}{2}} & 0 \\ 0 & dy_{\ell+\frac{1}{2}} \end{bmatrix}^{-1} \begin{bmatrix} D_{xx} & D_{xy} \\ D_{yx} & D_{yy} \end{bmatrix} \begin{bmatrix} dx_{k+\frac{1}{2}} & 0 \\ 0 & dy_{\ell+\frac{1}{2}} \end{bmatrix}^{-1} \begin{bmatrix} -1 & 1 & -1 & 1 \\ -1 & -1 & 1 & 1 \end{bmatrix} \right. \\
&\left. + \begin{bmatrix} 1 \\ -1 \\ -1 \\ 1 \end{bmatrix} \frac{\operatorname{tr}\left(\begin{bmatrix} dx_{k+\frac{1}{2}} & 0 \\ 0 & dy_{\ell+\frac{1}{2}} \end{bmatrix}^{-1} \begin{bmatrix} D_{xx} & D_{xy} \\ D_{yx} & D_{yy} \end{bmatrix} \begin{bmatrix} dx_{k+\frac{1}{2}} & 0 \\ 0 & dy_{\ell+\frac{1}{2}} \end{bmatrix}^{-1} \right)}{3} \begin{bmatrix} 1 & -1 & -1 & 1 \end{bmatrix} \right\} \\
&\quad \begin{bmatrix} \upsilon^{n+1}_{k,\ell} + \upsilon^n_{k,\ell} \\ \upsilon^{n+1}_{k+1,\ell} + \upsilon^n_{k+1,\ell} \\ \upsilon^{n+1}_{k,\ell+1} + \upsilon^n_{k,\ell+1} \\ \upsilon^{n+1}_{k+1,\ell+1} + \upsilon^n_{k+1,\ell+1} \end{bmatrix} \frac{dx_{k+\frac{1}{2}}\, dy_{\ell+\frac{1}{2}}}{4} \\
&\equiv \mathbf{M}_{k+\frac{1}{2},\ell+\frac{1}{2}} \begin{bmatrix} \upsilon^{n+1}_{k,\ell} - \upsilon^n_{k,\ell} \\ \upsilon^{n+1}_{k+1,\ell} - \upsilon^n_{k+1,\ell} \\ \upsilon^{n+1}_{k,\ell+1} - \upsilon^n_{k,\ell+1} \\ \upsilon^{n+1}_{k+1,\ell+1} - \upsilon^n_{k+1,\ell+1} \end{bmatrix} + \mathbf{K}_{k+\frac{1}{2},\ell+\frac{1}{2}} \frac{\triangle t^{n+\frac{1}{2}}}{2} \begin{bmatrix} \upsilon^{n+1}_{k,\ell} + \upsilon^n_{k,\ell} \\ \upsilon^{n+1}_{k+1,\ell} + \upsilon^n_{k+1,\ell} \\ \upsilon^{n+1}_{k,\ell+1} + \upsilon^n_{k,\ell+1} \\ \upsilon^{n+1}_{k+1,\ell+1} + \upsilon^n_{k+1,\ell+1} \end{bmatrix}
\end{aligned}
$$

The finite element equations at internal nodes then take the form

$$
(q_{00})_{k+\frac{1}{2},\ell+\frac{1}{2}} + (q_{10})_{k-\frac{1}{2},\ell+\frac{1}{2}} + (q_{01})_{k+\frac{1}{2},\ell-\frac{1}{2}} + (q_{11})_{k-\frac{1}{2},\ell-\frac{1}{2}} = 0\ . \tag{5.8}
$$

In general the sum, over elements within the computational domain, of the quadratures associated with a given node is zero. The mass matrix $\mathbf{M}$ and stiffness matrix $\mathbf{K}$ in equation (5.4) are formed by combining the elementwise mass and stiffness matrices from equations (5.7) into the finite element equations (5.8).

5.4. Weak Form with Sub-Domains. In AMR, we will use a hierarchy to define our computational grid. Our AMR algorithm will be recursive, always working with at most two grids in any region of space. Thus it will suffice locally to assume that our computational domain Ω satisfies $\overline{\Omega} = \overline{\Omega_f} \cup \overline{\Omega_c}$ where Ω_f and Ω_c are disjoint open sets (the fine and coarse domains). Let $S = \partial\Omega_f \cap \partial\Omega_c$ be the interface between the two domains, and let $\mu_f = \mathbf{n}_f \cdot D\nabla_x V$ be the diffusive flux, where $\mathbf{n}_f$ is the unit outer normal on Ω_f. We can introduce a Lagrange multiplier λ_f to force continuity of the solutions on the two domains, and develop the following weak formulation of the diffusion equation:

$$\forall w_f \in H^1(\Omega_f) \quad \frac{d}{dt}\int_{\Omega_f} w_f V_f \, dx + \int_{\Omega_f} \nabla_x w_f \cdot D\nabla_x V_f \, dx - \int_S w_f \mu_f \, dx = 0 \tag{5.9a}$$

$$\forall w_c \in H^1(\Omega_c) \quad \frac{d}{dt}\int_{\Omega_c} w_c V_c \, dx + \int_{\Omega_c} \nabla_x w_c \cdot D\nabla_x V_c \, dx + \int_S w_C \mu_f \, dx = 0 \tag{5.9b}$$

$$\forall \lambda_f \in H^{-\frac{1}{2}}(S) \quad \int_S \lambda_f(-V_f + V_c) \, dx = 0 \tag{5.9c}$$

If we choose finite dimensional subspaces $\mathcal{M}_f \subset H^1(\Omega_f)$, $\mathcal{M}_c \subset H^1(\Omega_c)$ and $\Lambda_f \subset H^{-\frac{1}{2}}(S)$, we can use the weak formulation on sub-domains to develop fully discrete finite element equations. The resulting fully discrete system will have the form

$$\begin{aligned}
&(\mathbf{M}_f + \mathbf{K}_f \frac{\triangle t^{n+\frac{1}{2}}}{2})\mathbf{V}_f^{n+1} - \mathbf{C}_f \mathbf{m}_f^{n+\frac{1}{2}} \triangle t^{n+\frac{1}{2}} = (\mathbf{M}_f - \mathbf{K}_f \frac{\triangle t^{n+\frac{1}{2}}}{2})\mathbf{V}_f^n \\
&(\mathbf{M}_c + \mathbf{K}_c \frac{\triangle t^{n+\frac{1}{2}}}{2})\mathbf{V}_c^{n+1} - \mathbf{C}_c \mathbf{m}_c^{n+\frac{1}{2}} \triangle t^{n+\frac{1}{2}} = (\mathbf{M}_c - \mathbf{K}_c \frac{\triangle t^{n+\frac{1}{2}}}{2})\mathbf{V}_c^n \\
&- \mathbf{C}_f^\top \mathbf{V}_f^{n+1} + \mathbf{C}_c^\top \mathbf{V}_c^{n+1} = 0
\end{aligned}$$

We will refer to the entries of $\mathbf{m}_f^{n+\frac{1}{2}}$ as the mortars [9].

In 1D, we will take the finite-dimensional subspace Λ_f of Lagrange multipliers to be the set of all scalar values at x_i. Suppose that $x_i = x_I \in S$ is a node shared by the fine and coarse domains, with the fine domain to the left of the coarse. With piecewise linear elements, our discrete equations take the form

$$\begin{aligned}
&(q_1)_{i-\frac{1}{2}} - \mu_i = 0 \, , \\
&(q_0)_{I+\frac{1}{2}} + \mu_i = 0 \, , \\
&- \upsilon_i + \upsilon_I = 0 \, .
\end{aligned}$$

The quadratures q_0, q_1 in these equations were defined in equation (5.5). In addition, we have incorporated the time step $\triangle t^{n+\frac{1}{2}}$ into the mortar μ_i. The third equation enforces continuity between the coarse and fine grid solutions, and the other two equations suggest that we define the coarse grid mortar at node $x_I = x_i \in S$ in terms of the fine mortar by

$$\mu_I = \mu_i \, .$$

These equations say that the composite grid equation at $x_i = x_I \in S$ is

$$(q_0)_{I+\frac{1}{2}} + \mu_I = 0 \text{ where } \mu_I = \mu_i = (q_1)_{i-\frac{1}{2}} \, .$$

Note that $(q_0)_{I+\frac{1}{2}}$ is a linear function of the coarse grid unknowns, and $(q_1)_{i-\frac{1}{2}}$ is a linear function of the fine grid unknowns and the coarse interface unknown υ_I (since continuity requires that $\upsilon_i = \upsilon_I$). Similar linearity conditions apply at other grid nodes. In general, the composite grid equations have the form of a linear system

$$\begin{bmatrix} \mathbf{A}_{ff} & \mathbf{A}_{fc} \\ \mathbf{A}_{cf} & \mathbf{A}_{cc} \end{bmatrix} \begin{bmatrix} \mathbf{u}_f \\ \mathbf{u}_c \end{bmatrix} = \begin{bmatrix} \mathbf{b}_f \\ \mathbf{b}_c \end{bmatrix}$$

where $\mathbf{u}_f$ is the vector of unknowns on the fine grid, and $\mathbf{u}_c$ is the vector of unknowns on the coarse grid.

In 2D, we will take Λ_f to be the set of all discontinuous piecewise linear functions on the fine mesh restricted to S. Our basis functions in Λ_f will be chosen to be dual to the piecewise linear basis functions in $\mathcal{M}_f$, restricted to S. As an example, suppose that S consists of a line segment parallel to the y-axis. Then if (x_i, y_j) is a boundary node, we want

$$\int_{y_0}^{y_N} V_{i,j}(x_i,y)\lambda_{j'}(y)\,dy = \frac{1}{2}\delta_{j,j'}\begin{cases} y_{j+1}-y_{j-1}, & 0<j<N \\ y_1 - y_0, & j=0 \\ y_N - y_{N-1}, & j=N \end{cases}$$

We will break S into intervals associated with the fine mesh, and assume that fine mortars at the endpoints of S are identical with the coarse grid values there. In this case, it is easy to see that the dual basis functions at nodes j interior to S are

$$\lambda_j(y) = \begin{cases} 2 - 3\frac{y-y_j}{y_{j+1}-y_j}, & y_j < y < y_{j+1} \\ 2 + 3\frac{y-y_j}{y_j - y_{j-1}}, & y_{j-1} < y < y_j \\ 0, & \text{otherwise} \end{cases}$$

The dual basis functions at the end nodes of S are the indicator functions for the end elements. With this choice for the basis functions in Λ_f, we find that $\mathbf{C}_f$ is diagonal.

Suppose that Ω_f is to the left of S at fine node (x_i, y_j), and that the coarse domain lies to the right of coarse node (x_I, y_J), where $x_i = x_I$. Also suppose that exactly two fine grid sides of equal length are contained in each coarse grid side within S. Then our discrete equations are

$$\begin{aligned} &(q_{11})_{i-\frac12,j-\frac12} + (q_{10})_{i-\frac12,j+\frac12} - \mu_{i,j} = 0 \\ &(q_{01})_{I+\frac12,J-\frac12} + (q_{00})_{I+\frac12,J+\frac12} + \frac{1}{2}\mu_{i,j-1} + \mu_{i,j} + \frac{1}{2}\mu_{i,j+1} = 0\ , \text{ where } y_j = y_J \\ &-v_{i,j} + \begin{cases} v_{I,J}, & y_j = y_J \\ \frac12(v_{I,J} + v_{I,J+1}), & y_{j+1} = y_J \end{cases} = 0\ . \end{aligned}$$

Here, we have incorporated the time step $\triangle t^{n+\frac12}$ and the mesh length into the mortars $\mu_{i,j}$. Similar equations hold at corners of the domains, or at physical boundaries. In general, the continuity condition implies that the fine grid nodal values on S are equal to the restriction of the coarse basis functions to S; in the current example,

$$v_{i,j} = \begin{cases} v_{I,J}, & y_j = y_J \\ \frac12(v_{I,J} + v_{I,J+1}), & y_{j+1} = y_J \end{cases}$$

The other equations imply that at a coarse node in S we should define the coarse grid mortar to be the fine grid mortar at that same node, plus half the sum of the fine grid mortars at neighboring fine grid nodes in S. In the current example, if $(x_i, y_j) = (x_I, y_J)$ then

$$\mu_{I,J} = \frac12\mu_{i,j-1} + \mu_{i,j} + \frac12\mu_{i,j+1}\ .$$

These give us equations to determine fine grid boundary values from coarse grid values, and to determine coarse grid mortars from fine grid mortars. These equations say that the composite grid equation at $(x_i, y_j) = (x_I, y_J) \in S$ is

$$\begin{aligned} &(q_{01})_{I+\frac12,J-\frac12} + (q_{00})_{I+\frac12,J+\frac12} + \mu_{I,J} = 0 \text{ where} \\ &\mu_{I,J} = \frac12\mu_{i,j-1} + \mu_{i,j} + \frac12\mu_{i,j+1} \\ &\quad = \frac12\big((q_{11})_{i-\frac12,j-\frac32} + (q_{10})_{i-\frac12,j-\frac12}\big) + (q_{11})_{i-\frac12,j-\frac12} + (q_{10})_{i-\frac12,j+\frac12} \\ &\quad + \frac12\big((q_{11})_{i-\frac12,j+\frac12} + (q_{10})_{i-\frac12,j+\frac32}\big) \end{aligned}$$

Note that the coarse quadratures are linear functions of the coarse grid unknowns, and the fine quadratures are linear functions of the fine grid unknowns and the coarse interface unknowns. (since continuity requires that the fine grid interface solution is the restriction of the coarse solution). Similar linearity conditions apply at other grid nodes. Again, the composite grid equations have the form

$$\begin{bmatrix} \mathbf{A}_{ff} & \mathbf{A}_{fc} \\ \mathbf{A}_{cf} & \mathbf{A}_{cc} \end{bmatrix} \begin{bmatrix} \mathbf{u}_f \\ \mathbf{u}_c \end{bmatrix} = \begin{bmatrix} \mathbf{b}_f \\ \mathbf{b}_c \end{bmatrix}$$

where $\mathbf{u}_f$ is the vector of unknowns on the fine grid, and $\mathbf{u}_c$ is the vector of unknowns on the coarse grid. The first block equation in this system represents the finite element equations at fine grid unknowns, and the second block equation represents the finite element equations at coarse grid nodes, include nodes in the coarse-fine interface S.

6. Adaptive Mesh Refinement. There are various forms of adaptive mesh refinement (AMR), ranging from static refinement [1, 51] to dynamic dynamic refinement in space [13, 14, 18, 23, 24, 25, 26, 27, 28, 29] and to dynamic refinement in both space and time [6, 8, 16, 33, 57]. Our approach follows, as closely as possible, the basic principles developed by Berger and Colella [6, 8].

In this paper, we will describe an AMR algorithm that dynamically selects refinement in both space and time. It will use a nested hierarchy of grids, each selecting appropriate timesteps chosen for needs of local accuracy and synchronization with coarser computations. The grid at a given level of refinement within the hierarchy will be a union of logically rectangular arrays of grid cells. These rectangular arrays of grid cells are called patches, and form the basic organizational unit for computation. Communication between patches is regular and small compared to the work within the patch, so the algorithm extends nicely to distributed computation.

We assume that we are given some initial coarse mesh. We will call this mesh the coarsest level of refinement, and denote it by $\mathcal{L}_0$. Finer levels $\mathcal{L}_\ell$ for $\ell > 0$ are defined recursively from the coarser level $\mathcal{L}_{\ell-1}$.

We assume that each level $\mathcal{L}_\ell$ consists of a finite array of patches, each of which is a logically rectangular array of cells. Here "logically rectangular" means that the array of cells can be mapped to a rectangular grid by a continuous coordinate transformation. The grid patches themselves can be non-rectangular in space; we only require that each patch be rectangular as a data array.

We assume that if a coarse cell is refined in any part of its physical space, then it is refined everywhere. As a result, the boundary of any fine patch on level $\mathcal{L}_\ell$ coincides with the boundary of a logically rectangular array of coarse grid cells on level $\mathcal{L}_{\ell-1}$. This assumption can restrict the grid generation for curvilinear grids; see [4] for an alternative.

We assume that we are given an integer refinement ratio r. Whenever a coarse cell on level $\mathcal{L}_\ell$ is refined, it is subdivided into r cells in each logical coordinate direction on level $\mathcal{L}_{\ell+1}$. The refinement ratio is a power of 2, and is commonly chosen to be 2 or 4. Note that the assumptions of a fixed refinement ratio and of grid alignment imply that on any level $\mathcal{L}_\ell$ with $\ell > 0$, in any patch the number of cells in any coordinate direction is an integer multiple of the refinement ratio.

We assume that the union of fine patches on level $\mathcal{L}_{\ell+1}$ is contained in the interior of the union of coarse patches on level $\mathcal{L}_\ell$. However, an individual fine patch is not required to lie inside any single coarse patch. Note that this assumption implies that the coarsest level $\mathcal{L}_0$ must completely cover the entire physical domain. This in turn implies that we must be able to provide a logically rectangular grid on the coarsest level. This assumption can restrict the range of application of this form of AMR. For modifications of the AMR technique that extend to more general problem geometry, see [5, 7].

After we advance the data on patches in a coarse level $\mathcal{L}_\ell$ to some time, we assume that the data on patches in a finer level $\mathcal{L}_{\ell+1}$ are advanced by as many timesteps as required by stability and accuracy to reach exactly the same time as the coarser level $\mathcal{L}_\ell$. This assumption implies that coarse patches are integrated before fine patches; it also implies that the timestepping algorithm must be applied recursively within each timestep on all but the finest level.

6.1. Timestep Selection for AMR. Since we are using stiffly stable integrators for reaction and diffusion, there is no *a priori* choice for the timestep on a given grid level. Since we are moving the mesh during AMR, however, we do need to make sure that the dynamical features requiring refinement do not move off the fine grid region before the next regridding event.

The Luo-Rudy reaction-diffusion system can generate traveling waves with reaonably well-determined speeds. From numerical experiments, we have determined the front speed to be $v = 6.6$ cm / milli-second for our model parameters. Thus, on any level of mesh refinement, we require $\triangle t \leq dx/v$. In order to enforce synchronization between fine and coarse computations, we also require the timestep on a fine level $\mathcal{L}_{\ell+1}$ to be an integer multiple of the timestep on its coarser level $\mathcal{L}_\ell$. If the finer level can be refined, then the number of timesteps on the fine level must also be an integer multiple of 2, so that we can perform Richardson error estimation for regridding, if necessary.

In performing the integration routines recursively, first we determine the initial timestep size as described previously in section 6.1. Then we loop over timesteps until we synchronize with the coarser level, if it exists. Within each timestep, we use operator splitting to integrate the reaction and diffusion equations; this is described below in section 6.3. Afterward, we use recursion to repeat the process on the finer levels. At appropriate times, we regrid the finer levels as described below in section 6.2. Finally, we determine the next timestep size.

6.2. Regridding. Because we are interested in solving time-dependent problems, we allow the mesh refinement to move in time. We use an error estimation procedure (described below) to determine where the unacceptably large errors occur on a given level. By using an error estimator, rather than a gradient detector, we are able to place mesh refinement where discontinuities in the variables are about to form, or where the algorithm is not able to produce second-order accuracy for some other reason, such as a lack of smoothness in the equation of state.

Let us describe how grid cells are selected for refinement. First, cells in the patches belonging to some given coarse level $\mathcal{L}_\ell$ are tagged if their global integration error is too large. This procedure uses both Richardson extrapolation to estimate the local truncation error in the integration, and a simple device to estimate the number of timesteps to be performed on this level of refinement. This error estimation procedure is a standard procedure in the numerical integration of ordinary differential equations [19]. Suppose that at each timestep in cell i we commit an error of magnitude ϵ_i (principally the local truncation error); further, suppose that the computation permits a bound M on the growth of these errors. Note that the Crank-Nicolson scheme finite difference equation (5.4) shows that the 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 flow and sufficiently fine mesh. Then the error $e_i^{(n)}$ in cell i at step n satisfies

$$e_i^{(0)} \leq \epsilon_i \ , \ e_i^{(n)} \leq \epsilon_i + M e_i^{(n-1)} \text{ for } n > 1.$$

An argument by induction shows that

$$e_i^{(n)} \leq \epsilon_i \sum_{j=0}^{n-1} M^j = \epsilon_i \frac{M^n - 1}{M - 1}.$$

If $M \approx 1 + \mu\epsilon_i$ is close to one, then for small n the error bound will be approximately $n\epsilon_i$. Suppose that the local truncation error satisfies

$$\epsilon_i = C \Delta t^{k+1} \ ,$$

where k is the expected global order of the scheme. (Note that our spatial and temporal error orders are equal.) Then the error in taking one coarse step of size $\rho\Delta t$ is

$$e_{i,c}^{(n)} \approx C \rho^{k+1} \Delta t^{k+1}.$$

On the other hand, if we take ρ fine timesteps of size Δt, the error is

$$e_{i,f}^{(n)} \approx C \rho \Delta t^{k+1}.$$

If $\rho > 1$, then this allows us to estimate the local error of a fine timestep by

$$\epsilon_i \approx \frac{e_{i,c}^{(n)} - e_{i,f}^{(n)}}{\rho^{k+1} - \rho} = \frac{w_{i,c}^{(n)} - w_{i,f}^{(n)}}{\rho^{k+1} - \rho},$$

where w is the quantity being monitored for errors. This gives us a computable estimate for the local truncation error. The accumulated error over N timesteps can be estimated by multiplying the local error by the anticipated number of timesteps N. We can approximate $N \approx \frac{L}{c\Delta t}$, where L is some length scale associated with the problem, c is the speed of the reaction front (see section 6.1), and Δt is the current timestep. As a result, cell i is tagged for refinement if the relative error satisfies

$$\frac{\mid w_{i,c}^{(n)} - w_{i,f}^{(n)} \mid}{\max_j \mid w_{j,f}^{(n)} \mid} \frac{L}{(\rho^{k+1} - \rho)c\Delta t} > tolerance.$$

In the computations described later in this paper, we always chose the tolerance to be 0.1.

Note that error estimation is performed on pseudo-patches that potentially lie in index spaces between the current level of refinement and the next coarser level. This is because the pseudo-patches are coarsened by a factor of the regrid interval $\rho = 2$, and mesh refinement uses an integer multiple $r = 2$ or 4 of the regrid interval. This reduces the work in comparing the errors, compared to estimating the error on the current level and something even finer.

It is important to consider the implementation of this error estimation strategy on a recursively refined mesh. Note that errors on coarse and fine meshes are estimated at different times, namely at one step forward on each individual level. At first glance, the fact that these times are different would appear to be undesirable. However, the alternative of comparing errors on all levels at the same time actually leads to much wasted work, and larger refined regions. This is because the error estimation on the coarse mesh places the refined cells where the disturbance will be moving, plus buffer cells. Thus it is only necessary to buffer by $\rho - 1$ cells (where ρ is the regrid interval), since that is the number of timesteps that will be taken between the times when the errors are estimated, and when the mesh will next be moved. If the errors had been computed at the same times, then it would be necessary to buffer by ρ cells on each level. Furthermore, the error estimation would have to proceed through more than one timestep, with recursive calls to integration on finer levels in order to provide data for finer grids. Since the mesh is going to be moved, this is extra work being performed for data that are only going to be discarded.

6.3. Advancing the Data. When we advance the data at a given level of refinement $\mathcal{L}_\ell$, we have available to us a grid hierarchy consisting of the current level of refinement and all coarser levels $\mathcal{L}_{\ell-1}, \ldots, \mathcal{L}_0$. We need to perform the steps of operator splitting for the reaction-diffusion system on level $\mathcal{L}_\ell$. This means that we react for a half-step, solve the diffusion equation for a full step, and then react for a half-step. This process, together with recursive integration of finer levels and infrequent regridding, continues until we have completed that number of steps required for synchronization with a coarser level $\mathcal{L}_{\ell-1}$.

There are important issues to examine in this process. One issue regards how to handle the initial reaction on the composite grid; this step provides the initial data for the diffusion step. We discuss the details of this initial reaction step in section 6.3.1. Another issue concerns the iterative solution of the linear system for the diffusion; this computation is discussed in section 7. Note that this linear system involves unknowns from patches on levels $\mathcal{L}_\ell, \ldots, \mathcal{L}_0$. A third issue involves the relationship of the second reaction to the synchronization of fine data with coarse; this is discussed in section 6.3.2.

6.3.1. Initial Operator-Split Reaction. We expect the error estimation process to select refinement in the neighborhoods of the strong reaction front. This is because the stiff reaction in the Luo-Rudy problem is not near steady-state there, so the errors in integrating the reactions are largest in that region. Away from the reaction front, we expect the potential to be varying slowly near equilibrium values for the reaction; both the diffusion and the reaction will involve small changes in the solution of the equations.

This suggests that we can use time interpolation on coarser levels $\mathcal{L}_{\ell-1}, \ldots, \mathcal{L}_0$ to determine the initial data for the linear system on those levels. We admit that this initial data should be the result, via operator splitting, of the current (fine) half-step of reaction; the effect of the time interpolation will be to approximate the result at the half-time of both reaction and diffusion on the coarse grid. However, in regions away from the reaction front, the fast reactions are nearly at steady state and the diffusion is relatively small. As a result, the difference in computational results between the two choices of initial data is negligible, while the difference in computational cost may be substantial for more complicated heart models.

Whenever we take a timestep on level $\mathcal{L}_\ell$, we use the SDIRK scheme to compute the results after a half-step of reaction. Note that this SDIRK scheme involves a local timestepping strategy (at each point in space) to guarantee a desired accuracy in the reaction integration. (See section 4.3).

6.3.2. Final Operator-Split Reaction and Synchronization. After we solve the linear system for the diffusion, we perform a second half-step of reaction in order to obtain overall second-order accuracy for the operator splitting process. This second reaction is performed on the current level $\mathcal{L}_\ell$ in the same way as it was performed before solving the diffusion equation. However, there are important differences in how the computations proceed on coarser levels $\mathcal{L}_{\ell-1}, \ldots, \mathcal{L}_0$.

In general, there is no need to perform the second reaction step on the coarser levels. These levels will use the results from their own coarse timesteps to determine, via time interpolation, the initial data for the next linear system for diffusion on the current level $\mathcal{L}_\ell$. The exception occurs when the current level reaches the same time as its coarser level $\mathcal{L}_{\ell-1}$, because it is necessary to synchronize the data between levels. At that point, it is necessary to perform a half-step of reaction on all synchronized levels.

We also need to adjust the coarse results near the boundary with the fine grid, in order to avoid boundary layers. The results of the reactions for synchronization replace the previous results from the reaction-diffusion-reaction step on the coarser levels. However, the iterative solution of the diffusion equation on the composite grid does not provide us with results for the solution on those portions of the coarse level $\mathcal{L}_{\ell-1}$ overlying the current level $\mathcal{L}_\ell$. (Recall that the multigrid algorithm computes *increments* to the solution on coarser levels.) The coarse grid results in regions overlying the current level are updated by the following process.

We want to replace the coarse results with appropriately coarsened results from the current level wherever the coarse and fine grids overlap. Given our nodal finite element data organization, this is easy. The coarse finite element space is nested in the fine space, so we can copy the fine results from fine grid nodes to coincidental coarse nodes. This corresponds to replacing coarse results with fine results on coarse particle paths at the end of the second reaction in operator splitting.

7. Iterative Linear Algebra. During each timestep on level $\mathcal{L}_\ell$, we solve a linear system for the Crank-Nicolson discretization of the diffusion equation. This linear system involves unknowns on levels $\mathcal{L}_\ell, \ldots, \mathcal{L}_0$. The initial data for this diffusion is provided by operator splitting on level $\mathcal{L}_\ell$, and by time interpolation on levels $\mathcal{L}_{\ell-1}, \ldots, \mathcal{L}_0$. All that remains in the description of the algorithm is the formulation and solution of the linear system for the diffusion equation on the hierarchical grid.

There are several parts to the description of this linear system. We use a preconditioned conjugate gradient iteration to solve the linear system on the composite grid. In section 5.4 we described the formulation of the linear system on a composite grid, composed of grid cells chosen from the grid hierarchy. These composite grid equations were related to a simple application of a popular technique called mortar finite elements. That discussion determines how we compute the residual and matrix-vector multiplies needed in the conjugate gradient algorithm.

In section 7.1 below we will describe a multiplicative domain decomposition process used to precondition the conjugate gradient iteration. If there were no local refinement, this multiplicative domain decomposition would be what is commonly called a multigrid algorithm.

7.1. Multiplicative Domain Decomposition. We will describe the basic ideas of multiplicative domain decomposition in this section. For more details on the convergence of the algorithm, the reader can consult [12, 13]. This multiplicative domain decomposition involves the same steps as multigrid: residual calculation, smoothing, restriction and prolongation. The residual calculation is completely determined by the composite grid equations in section 5.4. The prolongation involves injection of coarse grid finite element function values into fine grid interiors, and mortar continuity conditions on fine grid boundaries. The restriction involves the adjoints of the prolongation. The smoother operation is performed within a given level of refinement on a list of grid patches. It involves both non-overlapping additive domain decomposition between grid patches, and Gauss-Seidel iteration within a grid patch.

Let us consider the simple case where we have two domains $\Omega_f \subset \Omega_c$. For example, Ω_f may correspond to a fine grid in an adaptive approximation to the solution of some partial differential equation, and Ω_c may correspond to the coarse grid. Since the union of the grid patches in level$\mathcal{L}_{\ell+1}$ are contained in the interior of the union of the grid patches in the coarser level $\mathcal{L}_\ell$, the general algorithm can be reduced to recursive consideration of this two-level case. Specifically, there are no places in the composite grid where neighboring cells differ by more than one level of refinement.

Conceptually, we will think of the true unknown $\overline{u}_f$ and $\overline{u}_c$ on $\Omega_c \setminus \Omega_f$ as providing the best solution to the problem where it is available. Our composite grid linear system would then take the form

$$\begin{bmatrix} A_{ff} & A_{fc} \\ A_{cf} & A_{cc} \end{bmatrix} \begin{bmatrix} \overline{u}_f \\ \overline{u}_c \end{bmatrix} = \begin{bmatrix} b_f \\ b_c \end{bmatrix} . \tag{7.1}$$

Multiplicative domain decomposition is a particular iterative improvement algorithm for solving the composite grid equations (7.1). Without loss of generality, we will assume that our initial guess for the solution of the composite grid equations (7.1) is zero; if not, we apply the algorithm to the system with the same matrix and the initial residual as right-hand side. The multiplicative domain decomposition algorithm on level $calL_\ell$ with $\ell > 0$ takes the form

$$\begin{array}{rl}
\text{if on finest level, compute the initial fine residual} & r_f^{(0)} = -b_f \\
\text{pre-smooth on } \mathcal{L}_\ell & u_f^{(1)} = -S_f r_f^{(0)} \\
\text{compute the composite grid residual on} \mathcal{L}_\ell \text{ and } \mathcal{L}_{\ell-1} & \begin{bmatrix} r_f^{(1)} \\ r_c^{(0)} \end{bmatrix} = \begin{bmatrix} r_f^{(0)} \\ -b_c \end{bmatrix} + \begin{bmatrix} A_{ff} \\ A_{cf} \end{bmatrix} x_f^{(1)} \\
\text{restrict the residual from } \mathcal{L}_\ell \text{ to } \mathcal{L}_{\ell-1} & r_{c\leftarrow f} = R r_f \\
\text{recursively apply the algorithm on} \mathcal{L}_{\ell-1} & \begin{bmatrix} u_{cf} \\ u_c \end{bmatrix} = -\begin{bmatrix} V_{ff} & V_{fc} \\ V_{cf} & V_{cc} \end{bmatrix} \begin{bmatrix} r_{c\leftarrow f} \\ r_c^{(0)} \end{bmatrix} \\
\text{prolong the fine grid correction from } \mathcal{L}_{\ell-1} \text{ to } \mathcal{L}_\ell \text{ and update} & u_f^{(2)} = u_f^{(1)} + P u_{cf} \\
\text{compute the residual on } \mathcal{L}_\ell & r_f^{(2)} = r_f^{(0)} + \begin{bmatrix} A_{ff} & A_{fc} \end{bmatrix} u_f^{(2)} \\
\text{post-smooth on } \mathcal{L}_\ell & u_f = u_f^{(2)} - S_f^\top r_f^{(2)}
\end{array} \tag{7.2}$$

Here R is an inter-grid transfer operator that maps values on the fine grid to values on the coarse grid. We choose $R^\top$ to be injection from the coarse finite element space to the fine space.

On the coarsest level $\mathcal{L}_0$, the multiplicative domain decomposition algorithm takes the form

$$\text{solve } Au = r^{(0)} \tag{7.3}$$

This system of equations corresponds to a discretization of the diffusion equation on the entire domain. The residual $r^{(0)}$ was computed when the composite grid residual was evaluated on the next finer level $\mathcal{L}_1$. This linear system is relatively small, and can be solved effectively by a conjugate gradient iteration in multiple dimensions, or by a direct solve in one dimension.

In our multiplicative domain decomposition algorithm, the smoother S_f corresponds to approximating the solution of the linear equations for the differential equation on the subdomain Ω_f with appropriate boundary conditions. At physical boundaries, these boundary conditions should be specified. If Ω_f is smaller than the physical domain, then it is necessary to select appropriate internal boundary conditions for the definition of S_f. We will describe our choice for the smoother in section 7.2.

7.2. AMR Smoother. The only remaining piece of the iterative solution method is the smoother. As we have already mentioned, in one dimension we use a direct solver for the smoother. This means that on any grid patch that does not touch the boundary of the physical domain, we have Dirichlet boundary data coming from the coarse grid, and we solve directly for the interior unknowns. On any grid patch that does touch the physical boundary, we have Neumann data at that boundary and must solve for the solution at the boundary.

In multiple dimensions, the smoother must take a different form. We do not want to use a direct solver for the smoother in multiple dimensions, because the work is not proportional to the number of unknowns. Furthermore, the boundary nodes for a grid patch in level $\mathcal{L}_\ell$ may include nodes on the interface between levels $\mathcal{L}_\ell$ and $\mathcal{L}_{\ell-1}$, and nodes in the interior of the union of the patches on level $\mathcal{L}_\ell$. We need to choose a smoother that will perform well within the multiplicative domain decomposition algorithm, and still allow distributed computation.

We have chosen to use a Gauss-Seidel iteration at patch nodes other than at interior patch boundaries. At nodes on interior patch boundaries that are also interior to the union of the fine patches, we use a Jacobi iteration. At nodes on the interface with a coarser grid, a boolean mask tells us to ignore the correction from the Jacobi iteration.

Unlike the Gauss-Seidel iteration, Jacobi iteration does not depend on the order of processing of the unknowns. This means that if all patches on some level of refinement compute the same residual at shared nodes, then distributed Jacobi iterations will compute the same correction to the solution on all patches.

Our approach to the smoother corresponds to an additive domain decomposition, in the following sense. Consider the case in which there are two patches on level $\mathcal{L}_\ell$. We could order the equations and unknowns to obtain a linear system of the form

$$\begin{bmatrix} A_{11} & 0 & A_{1i} \\ 0 & A_{22} & A_{2i} \\ A_{i1} & A_{i2} & A_{ii} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_i \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_i \end{bmatrix}$$

Here u_1 is the vector of unknowns in the first patch, other than unknowns on the interface between the two patches. Similarly, u_2 is the appropriate vector of unknowns on the second patch, and u_i is the vector of unknowns on the interface between the two patches. We perform a block-Jacobi iteration on this system, where Gauss-Seidel iteration is used on the first two block equations (for the interior unknowns on each patch), and Jacobi iteration is used on the last block (for the unknowns on the interface between the patches).

8. Convergence Analysis of the V-cycle Preconditioner. In this section, following [10, 11, 15], we provide a convergence analysis of the multigrid V-cycle preconditioner used in the conjugate gradient iteration for the linear system (5.4). Let $\Omega_0 = \Omega$. For $l \geq 1$, let $\Omega_l \subset \Omega_{l-1}$ be the region where new nodes are added in the refinement level $\mathcal{L}_l$. Let $\mathcal{M}_l$ be the corresponding finite element space on the composite grid. Note that these spaces are nested, i.e.

$$\mathcal{M}_0 \subset \mathcal{M}_1 \subset \cdots \subset \mathcal{M}_L \equiv \mathcal{M}.$$

For each l, let $\widehat{\mathcal{M}_l}$ be the subspace of $\mathcal{M}_l$ defined by

$$\widehat{\mathcal{M}_l} = \{\phi \in \mathcal{M}_l \,|\, \phi = 0 \text{ on } \Omega \setminus \Omega_l\}.$$

For computational efficiency, the smoother in the V-cycle algorithm will operate on $\widehat{\mathcal{M}_l}$. The linear system (5.4) corresponds to the variational problem: *Find $u \in \mathcal{M}$ satisfying*

$$A(u,\phi) = (f,\phi) \quad \text{for all } \phi \in \mathcal{M},$$

where the bilinear form $A(\cdot,\cdot)$ on $\mathcal{M}$ is given by

$$A(u,v) = (u,v) + k\mathcal{B}(u,v) \quad \text{for all } u,v \in \mathcal{M}$$

with $k = \Delta t/2$ and the right hand side f is chosen appropriately.

The following linear operators are used in the multigrid algorithm. Let $A_l : \mathcal{M}_l \to \mathcal{M}_l$ and $\widehat{A}_l : \widehat{\mathcal{M}_l} \to \widehat{\mathcal{M}_l}$ be defined respectively by

$$(A_l u, v) = A(u,v) \quad \text{for all } u,v \in \mathcal{M}_l$$

and

$$(\widehat{A}_l u, v) = A(u,v) \quad \text{for all } u,v \in \widehat{\mathcal{M}_l}.$$

Let $S_l : \widehat{\mathcal{M}_l} \to \widehat{\mathcal{M}_l}$ be the smoother such as Richardson, Jacobi, and Gauss-Seidel smoothers. Its adjoint with respect to the inner product $(\cdot,\cdot)$ is denoted by $S_l^t : \widehat{\mathcal{M}_l} \to \widehat{\mathcal{M}_l}$. The operators $\widehat{Q}_l : \mathcal{M} \to \widehat{\mathcal{M}_l}$ and $Q_l : \mathcal{M} \to \mathcal{M}_l$ denote the L^2-orthogonal projection onto $\widehat{\mathcal{M}_l}$ and $\mathcal{M}_l$, respectively.

The V-cycle preconditioner B_L is given by the following recursive algorithm.

ALGORITHM 8.1. *Let $B_0 = A_0^{-1}$. For $l \geq 0$, assuming that $B_{l-1} : \mathcal{M}_{l-1} \to \mathcal{M}_{l-1}$ has been defined, $B_l : \mathcal{M}_l \to \mathcal{M}_l$ is defined as follows. Let $g \in \mathcal{M}_l$.*

1. *Pre-smoothing: Set $v' = \widehat{S}_l^t \widehat{Q}_l g$.*
2. *Correction: Set $v'' = v' + B_{l-1} Q_{l-1}(g - A_l v')$.*

3. Post-smoothing: Set $B_l g = v'' + \widehat{S}_l \widehat{Q}_l (g - A_l v'')$.

The analysis is based on the following three conditions. The first one is a stable approximation property of subspaces $\mathcal{M}_l$.

(A) There exist linear operators $\Pi_l : \mathcal{M} \to \mathcal{M}_l$ (not necessarily projectors) with $\Pi_L = I$ and range$(\Pi_l - \Pi_{l-1}) \subset \widehat{\mathcal{M}}_l$ such that

$$\|(\Pi_l - \Pi_{l-1})v\|^2 \le C_\pi \lambda_l^{-1} A(v,v), \qquad \text{for all } v \in \mathcal{M}, 1 \le l \le L \tag{8.1}$$

and

$$A(\Pi_l v, \Pi_l v) \le C'_\pi A(v,v), \qquad \text{for all } v \in \mathcal{M}, l \le l \le L. \tag{8.2}$$

Here, λ_l is the largest eigenvalue of the operator A_l.

The other two are smoothing conditions. Condition **(S.1)** guarantees that the smoothing operations S_l and S_l^t effectively reduce the components of the error corresponding to large eigenvalues of A_l. On the other hand, condition **(S.2)** requires the smoother to be properly scaled.

(S.1) Let $\overline{\widehat{S}}_l = \widehat{S}_l + \widehat{S}_l^t - \widehat{S}_l^t \widehat{A}_l \widehat{S}_l$. There exists a positive constant $\omega \in (0,1]$ not depending on L such that

$$\frac{\omega}{\lambda_l}\|v\|^2 \le (\overline{\widehat{S}} v, v), \qquad \text{for all } v \in \widehat{\mathcal{M}}_l, 1 \le l \le L.$$

(S.2) There exists a constant $\theta \in (0,2)$ not depending on $l = 1, 2, \cdots, L$ such that

$$A(\widehat{S}_l v, \widehat{S}_l v) \le \theta(\widehat{S}_l v, v), \qquad \text{for all } v \in \widehat{\mathcal{M}}_l.$$

The following generalization of Theorem 1 of [10] in the context of mesh refinement [15] is found in [11].

THEOREM 8.1 (Theorem 6.2, [11]). *Assume that Condition* **(A)** *holds and that the smoothers* S_l, *satisfy Conditions* **(S.1)** *and* **(S.2)**. *Then,*

$$0 \le A((I - B_L A_L)v, v) \le \left(1 - \frac{1}{C_L}\right) A(v,v), \qquad \textit{for all } v \in \mathcal{M},$$

or equivalently,

$$\frac{1}{C_L} A(v,v) \le A(B_L A_L v, v) \le A(v,v), \qquad \textit{for all } v \in \mathcal{M},$$

with

$$C_L = \left\{ \left(1 + (L-1)\frac{C_\pi}{\omega}\right)^{1/2} + \left(2(L-1)\frac{(1+C'_\pi)\theta}{2-\theta}\right)^{1/2} \right\}^2.$$

It is easy to see that the smoothing conditions **(S.1)** and **(S.2)** are satisfied by simple smoothers such as the Richardson smoother. We will verify condition **(A)**. Let $\widetilde{\mathcal{M}}_l$ be the finite element space corresponding to the grid obtained by a uniform refinement over the entire domain Ω on level $\mathcal{L}_l$ and $\widetilde{Q}_l : \mathcal{M} \to \widetilde{\mathcal{M}}_l$ be the L^2-orthogonal projection onto $\widetilde{\mathcal{M}}_l$. Let $\Omega_{L+1} = \emptyset$. For $1 \le l \le L$, define $\Pi_l : \mathcal{M} \to \mathcal{M}_l$ as in [10, 15] by

$$\Pi_l v = \begin{cases} v & \text{at nodes of } \mathcal{M}_l \text{ in the interior of } \Omega_{l+1}, \\ \widetilde{Q}_l v & \text{at remaining nodes of } \mathcal{M}_l. \end{cases}$$

By construction, it is easy to see that $\Pi_L = I$ and $(\Pi_l - \Pi_{l-1})v \in \widehat{\mathcal{M}}_l$.

The operator Π_l was used in [15, 10] to establish condition **(A)** for the bilinear form $\mathcal{B}(\cdot,\cdot)$ and we have

$$\mathcal{B}(\Pi_l v, \Pi_l v) \lesssim \mathcal{B}(v,v), \qquad \text{for all } v \in \mathcal{M}, 1 \le l \le L, \tag{8.3}$$

and

$$\mu_l \|(\Pi_l - \Pi_{l-1})v\|^2 \lesssim \mathcal{B}(v,v), \qquad \text{for all } v \in \mathcal{M}, 1 \le l \le L, \tag{8.4}$$

where μ_l be the largest eigenvalue of the operator $T_l : \mathcal{M}_l \to \mathcal{M}_l$ defined by

$$(T_l u, v) = \mathcal{B}(u,v), \qquad \text{for all } u, v \in \mathcal{M}_l.$$

Here and in the rest of this section, $A \lesssim B$ means that there is a constant c independent of the mesh size h or the time step size Δt or the number of refinement levels L such that $A \le cB$. In [10], it was also shown that

$$\|(I - \Pi_l)v\| \lesssim \|(I - \widetilde{Q}_l)v\|, \qquad \text{for all } v \in \mathcal{M} \tag{8.5}$$

We first consider (8.2). From triangle inequality and (8.5), we have

$$\|\Pi_l v\| \le \|v\| + \|(I - \Pi_l)v\| \lesssim \|v\|. \tag{8.6}$$

Combining this and (8.3) gives (8.2). Finally, since $\lambda_l = 1 + k\mu_l$, we have, from (8.4) and (8.6), that

$$\lambda_l \|(\Pi_l - \Pi_{l-1})v\|^2 \lesssim A(v,v), \qquad \text{for all } v \in \mathcal{M}, 1 \le l \le L.$$

This is (8.1) and we have established **(A)** for the bilinear form $A(\cdot,\cdot)$.

9. Numerical Results. We have considered 1D and 2D simulations of electrical wave propagation in a model heart. All computations used a refinement ratio of 4, second-order operator splitting and the Crank-Nicolson scheme for time integration. The diffusion coefficient (membrane conductance / membrane capacitance) as chosen to be the constant $D(x) = 1.25 \times 10^{-3}$. The heart tissue was 6 cm long in each coordinate direction.

For the first milli-second, we applied a simulus current of -100 micro-amps / square cm along a band of tissue within 1/40 of the tissue width from the left-hand side. In two dimensions, from time 315 to 316 milli-seconds we applied a second stimulus of -50 micro-amps / square cm in the lower left-hand quarter of the tissue. This generateed spiral waves in 2D, and interesting AMR.

In order to hasten the relaxation of the tissue after the initial stimulus, we have modified three parameters in the Luo-Rudy model. The fast sodium current, slow inward current and time-dependent potassium current were changed from the original model in (2.5c) (2.5d) and (2.5b) to be

$$\begin{aligned} I_{Na}(\mathbf{g}, V) =& 16\; m^3\; h\; j\; (V - 54.4)\;. \\ I_{si}(\mathbf{g}, [Ca]_i, V) =& 0.052\; d\; f\; (V - 7.7 + 13.0287\; \log([Ca]_i)) \\ I_{K_1}(\mathbf{g}, V) =& 0.423\; K_1\; (V + 87.26) \end{aligned}$$

The changes involve only the leading coefficient in the right-hand side expressions.

Our coarsest grid consisted of 20 grid cells in each coordinate direction. Without refinement, this grid is too coarse and causes pinning of the reaction front. As the mesh is refined, the reaction front gains speed until a limiting value is reached. With 10 to 12 grid cells inside the reaction front, the computed front speed is accurate to about 2 digits.

9.1. Numerical Results in 1D. The wave propagation was simulated for a total of 120 mili-seconds in 1D. This is slightly more than the amount of time it takes for the reaction front due to the initial stimulus to cross the tissue. After this time, the AMR algorithm removes all refinement, and takes large and efficient timesteps on the coarsest level $\mathcal{L}_0$ only. If we had run the comparisons between uniform grid and AMR simulations to later time, we would have made the uniform grid computations look comparatively worse. In figure 9.1 we show some computational results for uniform and AMR simulations in 1D. These show the jump in membrane potential propagating to the right, with good agreement between uniform and AMR computations.

In 1D it is possible to run very detailed uniform grid computations in a reasonable amount of time. A uniform grid computation with 5120 cells took 907 seconds, or slightly more than 15 minutes, on our Dec Alpha machine. The corresponding AMR simulation took 124 seconds, or slightly more than 2 minutes.

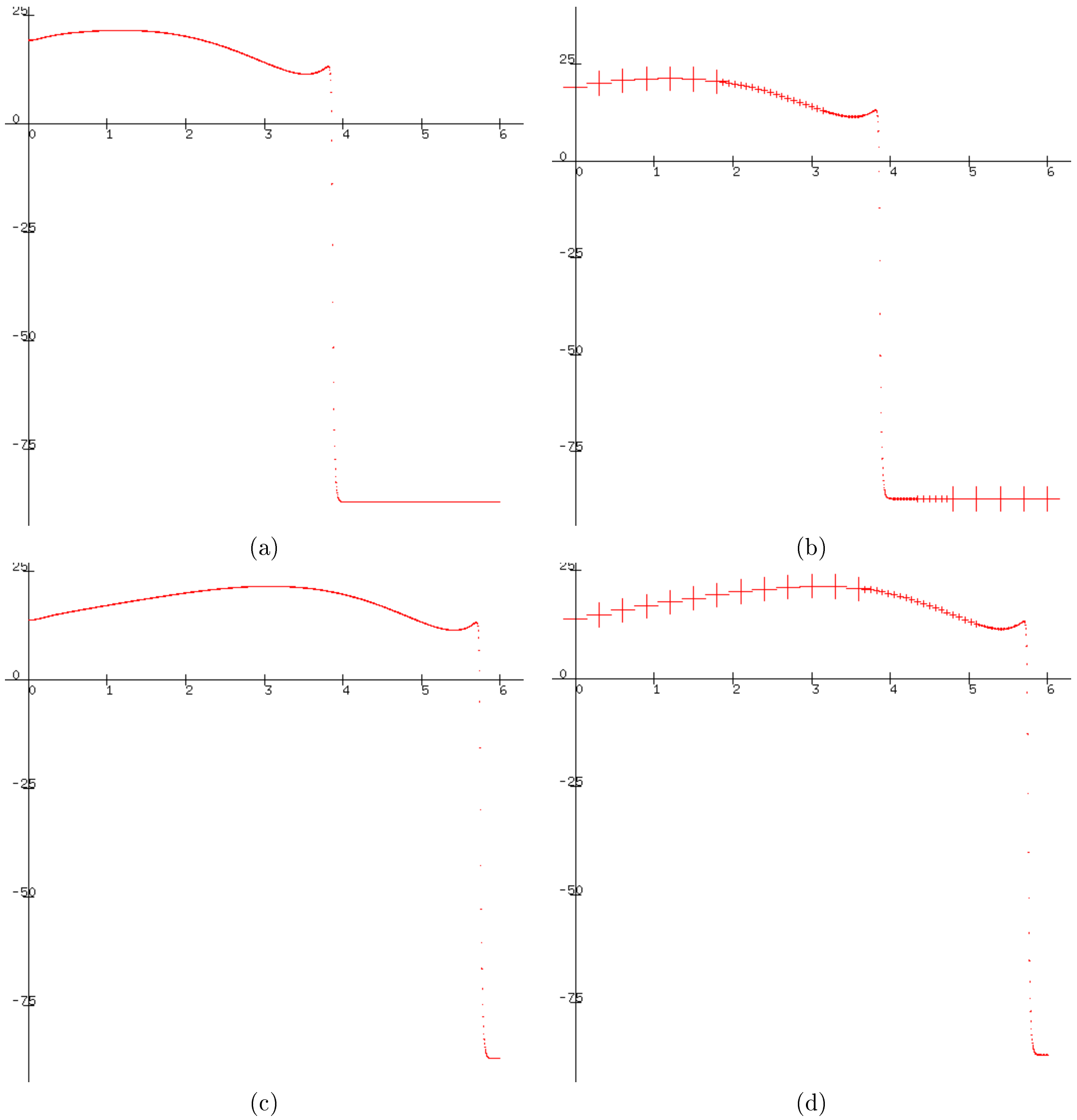

Fig. 9.1. *Membrane potential versus position. (a) Uniform grid of 1280 cells at 60 mili-seconds; (b) AMR with 4 levels at 60 mili-seconds; (c) Uniform grid of 1280 cells at 90 mili-seconds; (d) AMR with 4 levels at 90 mili-seconds.*

Without AMR, we expect that refinement by a factor of r will roughly cost a factor of r^2 more work. This is because the work per timestep is proportional to the number of grid cells, and the timestep is proportional to the cell width. (If we used explicit time integration for the diffusion as in [16], the timestep would have been portional to the cell width squared for numerical stability, and the work should roughly increase by a factor of r^3 as a result of refinement.) Thus, refinement by a factor of 4 should increase the cost of uniform grid computations by a factor of 16. In practice, we observed that the computational time increased by factors of 7.78 (from 20 to 80 cells), 11.17 (from 80 to 320 cells), 14.89 (from 320 to 1280 cells) and 17.50 (from 1280 to 5120 cells). The observed ratios can be larger than predicted because we change the tolerance for the reaction computations as we refine the mesh, as described in section 4.3. At low resolution, the computational time can be less than predicted because the reaction wave is not propagating properly (if at all), and the solution is not changing much from one timestep to the next.

The computational results for 1D uniform grid simulations show that most of the computational time is spent in integrating the reactions. For problems with at least 80 grid cells, 90% to 95% of computational time is spent in this task. It should be noted, however, that the solution of the linear system for the diffusion equation is especially simple in 1D, because we can use a tridiagonal system solver. We should not expect the reaction computations to dominate to such an extent in multiple dimensions.

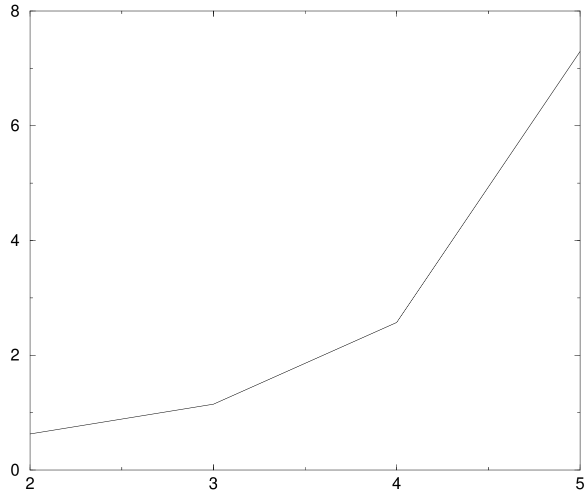

FIG. 9.2. *Speedup due to AMR versus levels of refinement in 1D. Level 0 involved 20 cells; refinement by a factor of 4.*

The computational results for 1D AMR computations show a different scaling with mesh refinement. Suppose that the propagating front were refined with a fixed number of grid cells normal to the front on each level of refinement. Then adding another level of refinement should roughly increase the computational work by a factor of the refinement ratio r, because the timestep on the fine grid would be decreased by that factor. In practice, we found that the computational time increased by a factor of 6.15 (from 2 to 3 levels), 6.63 (from 3 to 4 levels) and 6.16 (from 4 to 5 levels). Probably, these factors are greater than 4 because the tolerances for both the reactions and the iterative solution of the linear systems for the diffusion are tightened as we refine the mesh, as described in section 4.3. Nevertheless, the computational time increases much less rapidly with AMR than with uniform grid refinement. This indicates that for sufficiently refined computations, the AMR computations should take less computational time than the uniform grid computations.

The computational results for 1D AMR simulations show that about 1/3 of the computational time is spent in integrating the reactions, and about 2/3 of the total time in solving the linear systems for diffusion. About 12% of simulation time is spent in error estimation for regridding; this includes time for solving linear systems and for integrating reactions. In 1D, the linear system solves with AMR take relatively more work than uniform grid computations for two reasons. First of all, with AMR there is less work to do in integrating the reactions because there are fewer grid cells. Secondly, there is more work to do in the linear system solve, because the preconditioned conjugate gradient iteration using multiplicative domain decomposition for AMR cannot be performed as rapidly as a single tridiagonal system solve. This is in spite of that fact that though both algorithms (tridiagonal solve and conjugate gradients preconditioned by multiplicative domain decomposition) involve work proportional to the number of unknowns.

In figure 9.2 we show the speedup due to AMR, compared to uniform grid computations. With 80 grid cells on a uniform grid compared to 20 grid cells refined adaptively once by a factor of 4, we find that the AMR code runs slower than the uniform grid computation. However, if we compare a uniform grid of 5120 cells to AMR with 5 levels using a refinement ratio of 4, we find that the AMR computation runs a factor of 7.29 faster than the uniform grid computation.

9.2. Numerical Results in 2D. In 2D, the wave propagation was simulated for a total of 460 miliseconds. This is slightly more than the amount of time it takes for the spiral wave due to the second stimulus to circle around the tissue. After this time, the spiral wave dissipates and AMR removes all refinement. Between roughly 120 and 315 seconds in simulation time, the tissue is relaxing from the initial stimulus, and AMR is able to remove all refinement. Uniform grid computations waste a great deal of simulation time in this time interval.

In figure 9.3 we show some computational results for AMR computations in 2D. As a result of the first stimulus along the left-hand side of the domain, the initial jump in membrane potential propagates to the right, then the tissues relaxes. After the second stimulus in the lower left-hand quadrant of the tissue, a spiral wave moves clockwise around the domain. The contour plots use a hot-cold color map (blue for low values, red for high), with the grid patches drawn over the contours.

In 2D we are not able to perform calculations with more than 640 cells in each coordinate direction, because of the size of available memory. A uniform grid computation with 320 cells in each direction took 4863 seconds, or slightly more than 81 minutes, on our Dec Alpha machine. The corresponding AMR simulation using 5 levels and a refinement ratio of 2 took 2812 seconds, or slightly less than 47 minutes. AMR using 3 levels and a refinement ratio of 4 took 1113 seconds, or about 18.5 minutes. In general, it is more efficient to use a refinement ratio of 4 than a ratio of 2.

Without AMR, we expect that refinement by a factor of r will roughly cost a factor of r^3 more work. This is because the work per timestep is proportional to the number of grid cells (which multiplies the work by $O(r^2)$), and the timestep size is proportional to the cell width (which multiplies the work by $O(r)$). If we had used explicit time integration for the diffusion, the timestep would have been portional to the square of the cell width for numerical stability, and the work would roughly increase by a factor of r^4 as a result of refinement. Thus, refinement by a factor of 4 should increase the cost of uniform grid computations by a factor of 64, and refinement by a factor of 2 should increase the cost by a factor of 8. In practice, we observed that the computational time increased by factors of 5.34 (from 20 to 40 cells), 7.61 (from 40 to 80 cells), 8.28 (from 80 to 160 cells) and 17.50 (from 1280 to 5120 cells). With refinement by a factor of 4, we found that the uniform grid computational time increased by factors 40.7 (from 20 to 80 cells) and 60.5 (from 80 to 320 cells). We believe that the observed ratios were larger than predicted in some cases because we change the tolerance for the reaction computations as we refine the mesh, as described in section 4.3. At low resolution, the computational time can be less than predicted because the reaction wave is not propagating properly (if at all), and the solution is not changing much from one timestep to the next.

The computational results for 2D uniform grid simulations again show that most of the computational time is spent in integrating the reactions. For all of the 2D simulations (using 20 to 640 cells in each direction), 90% of computational time is spent in this task. This is in spite of the fact that we are using a simple conjugate gradient iteration (without preconditioning) to solve the uniform grid linear system. However, as the mesh is refined the cost of the reaction integrations descreases slightly, as the conjugate gradient iteration begins to take more iterations to converge.

The computational results for 2D AMR computations show a different scaling with mesh refinement. Suppose that the propagating front were refined with a fixed number of grid cells normal to the front on each level of refinement. Then adding another level of refinement should roughly increase the computational work by a factor of the refinement ratio r^2, because the timestep on the fine grid would be decreased by factor of r and we would use a factor of r more grid cells tangential to the front. (If we had used explicit time integration for the diffusion, as in [16], the work would be expected to scale proportional to r^3.) Thus with a refinement ratio of 2, we expect that the computational cost should increase by a factor of 4 with each new level of AMR. In practice, we found that the computational time increased by a factor of 4.62 (from 2 to 3 levels), 9.14 (from 3 to 4 levels) and 5.59 (from 4 to 5 levels). Probably, these factors are greater than 4 because the tolerances for both the reactions and the iterative solution of the linear systems for the diffusion are tightened as we refine the mesh, as described in section 4.3. Nevertheless, the computational time increases less rapidly with AMR than with uniform grid refinement. This indicates that for sufficiently refined computations, the AMR computations should take less computational time than the uniform grid computations.

The computational results for 2D AMR simulations show that about 3/4 of the computational time is spent in integrating the reactions, and about 1/4 of the total time in solving the linear systems for diffusion. In practice, the preconditioned conjugate gradient iteration worked very well, typically using 1 or 2 iterations to converge to the tolerance described in section 4.3. About 10% of simulation time is spent in error estimation for regridding; this includes time for solving linear systems and for integrating reactions.

In figure 9.4 we show the speedup due to AMR, compared to uniform grid computations. With 40 grid

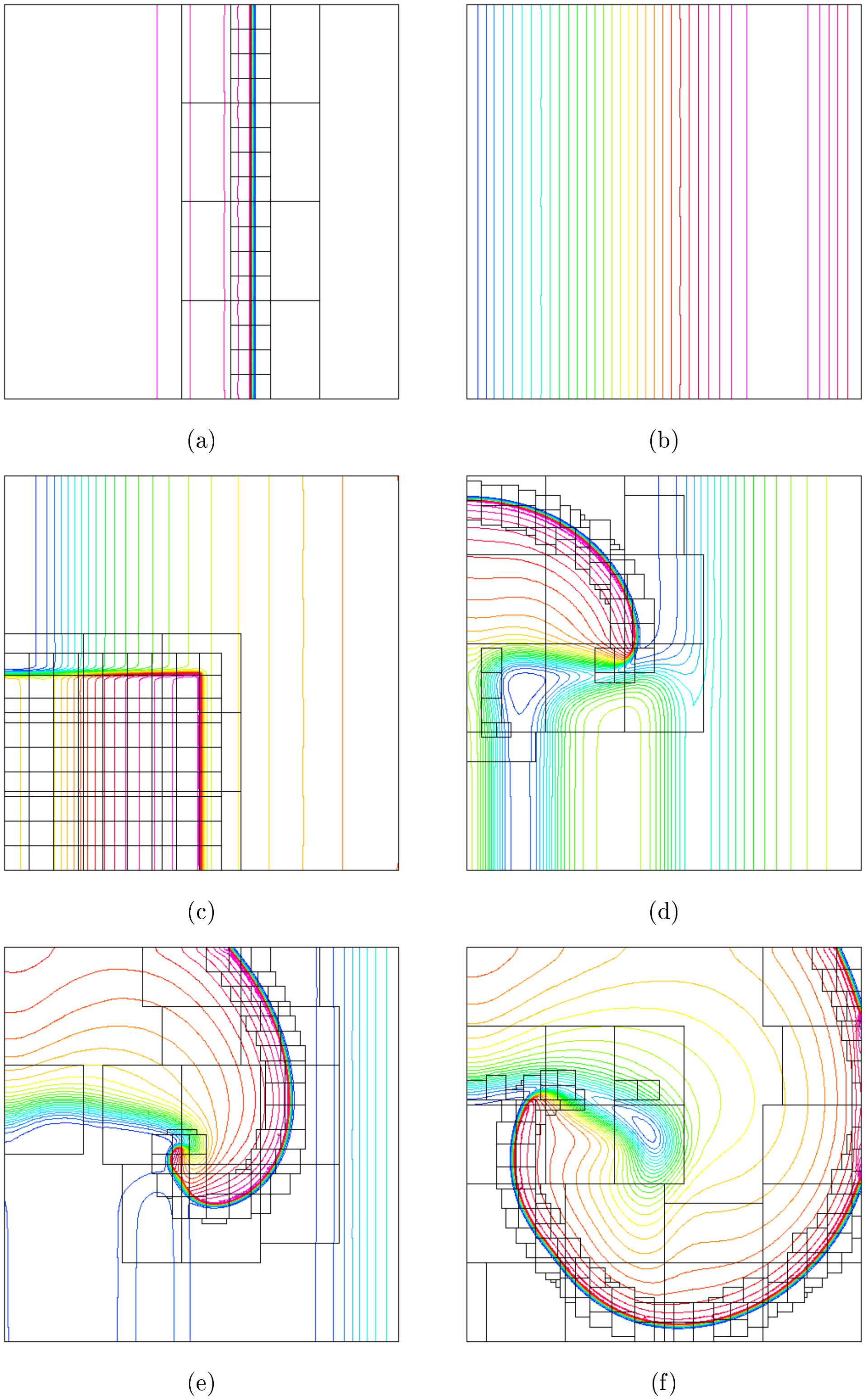

Fig. 9.3. *Contours of Membrane potential, AMR, 3 levels, 40 cells on coarsest level. (a) 60 mili-seconds; (b) 120 mili-seconds; (c) 318 mili-seconds; (d) 360 mili-seconds; (e) 390 mili-seconds; (f) 420 mili-seconds;*

cells in each direction on a uniform grid compared to 20 grid cells refined adaptively once by a factor of 2, we find that the AMR code runs slower than the uniform grid computation. However, if we compare a uniform grid of 320 cells in each direction to an AMR grid with 5 levels using a refinement ratio of 2, we find that the AMR computation runs a factor of 1.72 faster than the uniform grid computation. The speedup is significantly greater when AMR uses a refinement ratio of 4. For example, a uniform grid of 320 cells in each direction takes a factor of 4.35 more computational time than an AMR simulation using 3 levels refined from a coarse grid with 20 cells in each direction.

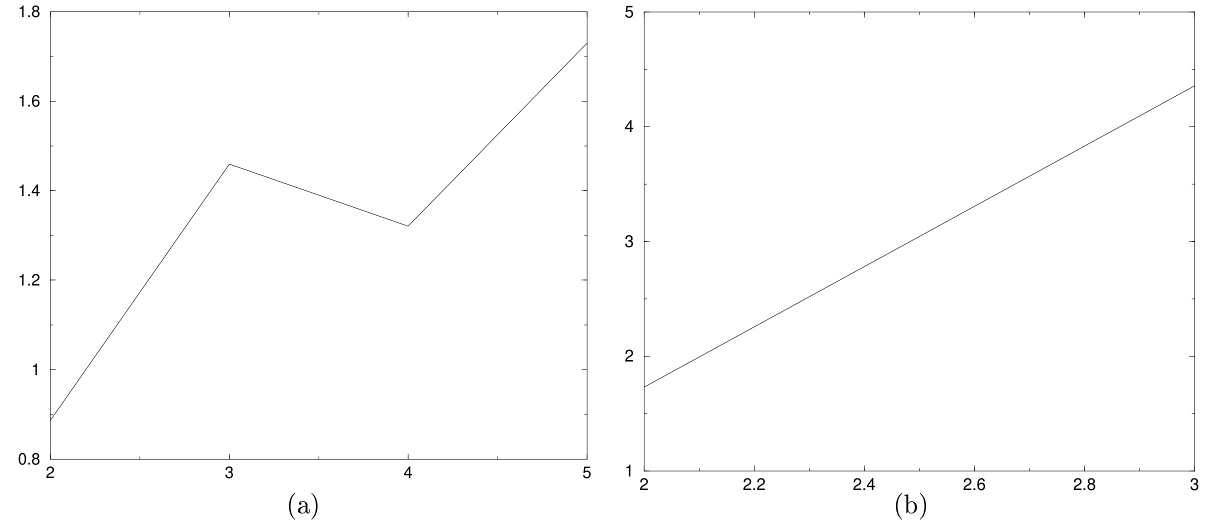

FIG. 9.4. *Speedup due to AMR versus levels of refinement in 2D. Level 0 involved 20 cells. (a) refinement by a factor of 2; (b) refinement by a factor of 4;*

10. Conclusions. Iterative solution of linear systems within AMR leads to significant additional complexity for the AMR code. Further, many of the details of this code implementation cannot be carried over to other AMR applications.

For example, we have previously developed iterative methods for solving linear systems arising from incompressible flow in porous media within AMR. In order to conserve fluid volume and treat random permeability via harmonic averaging, we used a non-conforming hybrid mixed finite element method combined with mortar elements. In that porous media application, the volume conservation naturally gave rise to fine grid unknowns as master to the coarse unknowns at interfaces between coarse and fine grids. In this application of AMR to a diffusion equation, we used a conforming finite element method, in which it was natural for the coarse grid unknowns on the interface to be master to the fine. This reversal of communication rippled through a fair amount of the multiplicative domain decomposition preconditioner.

In spite of the difficulties in code development, we are encouraged by the relative preformance advantage of AMR, when compared to uniform grid computations. Because of the difficulty in programming AMR for these applications, we have made the code publically available at

`http://www.math.duke.edu/~johnt/duke_amr.html`.

This software consists of eight libraries and four application examples. This code has been programmed for distributed computing; we will report on the success of that development in a later paper.

11. Acknowledgments. We would like to thank Kenneth Eaton and Craig Henriquez of Duke University for many helpful comments during the development of this application of AMR to the Luo-Rudy model.

REFERENCES

[1] R.E. Bank. A-posteriori error estimates, adaptive local mesh refinement and multigrid iteration. In *Proceedings of the 2nd European Conference on Multigrid Methods*, Cologne, October 1-4, 1985, 1986. W. Hackbusch and U, Trottenberg, eds.
[2] John P. Barach. A simulation of cardiac action currents having curl. *IEEE Trans. Biomed. Eng.*, 40:49–58, 1993.

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

[4] J.B. Bell, P. Colella, J. Trangenstein, and M. Welcome. Godunov methods and adaptive algorithms for unsteady flow. In *Proceedings of the 11th International Conference on Numerical Methods in Fluid Dynamics, June 27-July 1*, 1988.

[5] M. Berger and J. Melton. An accuracy test of a cartesian grid method for steady flow in complex geometries. In *Proceedings of the 5th International Conference on Hyperbolic Problems*, Stonybook, NY, June, 1994.

[6] M. J. Berger and P. Colella. Local adaptive mesh refinement for shock hydrodynamics. *J. Comp. Phys.*, 82:64–84, 1989.

[7] M. J. Berger and R.J. LeVeque. Stable boundary conditions for Cartesian grid calculations. *Comput. Syst. Engin.*, 1:305–31, 1990.

[8] M.J. Berger and J. Oliger. Adaptive mesh refinement for hyperbolic partial differential equations. *J. Comp. Phys.*, 53:484–512, 1984.

[9] C. Bernardi, Y. Maday, and A.T. Patera. A new nonconforming approach to domain decomposition: the mortar element method. In H. Brezis and J.L. Lions, editors, *Nonlinear Partial Differential Equations and Their Applications*, number 290 in Pitman Research Notes on Mathematics, pages 13–51, Harlow, UK, 1989. Longman Scientific&Technical.

[10] J. H. Bramble, J. E. Pasciak, J. Wang, and J. Xu. Convergence estimates for multigrid algorithms without regularity assumptions. *Math. Comp.*, 57:23–45, 1991.

[11] J. H. Bramble and X. Zhang. *Handbook of numerical analysis*, volume VII, chapter entitled The analysis of multigrid methods, pages 173–415. North-Holland, 2000. P. G. Ciarlet and J. L. Lions, eds.

[12] James H. Bramble. *Multigrid Methods*, volume 294 of *Pitman Research Notes in Mathematics*. Longman, London, 1993.

[13] James H. Bramble, Richard E. Ewing, Rossen R. Parashkevov, and Joseph E. Pasciak. Domain decomposition methods for problems with partial refinement. *SIAM J. Sci. Stat. Comput.*, 13:397–410, 1992.

[14] James H. Bramble, Richard E. Ewing, Joseph E. Pasciak, and Alfred H. Schatz. A preconditioning technique for the efficient solution of problems with local grid refinement. *Comput. Methods Appl. Mech. and Engrg.*, 67:149–159, 1988.

[15] James H. Bramble and Joseph E. Pasciak. New estimates for multilevel algorithms including the v-cycle. *Math. Comp.*, 60:447–471, 1993.

[16] Elizabeth M. Cherry, Henry S. Greenside, and Craig S. Henriquez. A space-time adaptive method for simulating complex cardiac dynamics. *Phys. Review Letters*, 84:1343–1346, 2000.

[17] M. Courtemanche and A. Winfree. Re-entrant rotating waves in a Beeler-Reuter based model of two-dimensional cardiac electrical activity. *Intl J Bifurcation and Chaos*, 1:431–44446, 1991.

[18] H.K. Dahle, M.S. Espedal, R.E Ewing, and O. Savareid. Characteristic adaptive subdomain methods for reservoir flow problems. *Numer. Meth. Partial Diff. Equations*, 6:279–309, 1990.

[19] G. Dahlquist and Å. Björck. *Numerical Methods*. Prentice-Hall, 1974. Translated by N. Anderson.

[20] K. Dekker and J.G. Verwer. *Stability of Runge-Kutta Methods for Stiff Nonlinear Differential Equations*. North-Holland, 1984.

[21] J.P. Drouhard and F.A. Roberge. The simulation of repolarization events of the cardiac Purkinje fiber action potential. *IEEE Trans. Biomed. Eng.*, 29:481–493, 1982.

[22] Emilia Entcheva, Natalia A. Trayanova, and Fritz J. Claydon. Patterns and mechanisms for shock-induced polarization in the heart: A bidomain analysis. *—TBME—*, 46:260–270, 1999.

[23] R. E. Ewing, R. D. Lazarov, T. F. Russell, and P. S. Vassilevski. Analysis of the mixed finite element method for rectangular Raviart-Thomas elements with local refinement. In T.F. Chan, R. Glowinski, J. Periaux, and O. Widlund, editors, *Domain Decomposition Methods for Partial Differential Equations*, pages 98–114, Philadelphia, Penn., 1990. SIAM.

[24] R.E. Ewing. *Adaptive Methods for Partial Differential Equations*, chapter Adaptive Grid Refinements for Transient Flow Problems, pages 194–205. SIAM, 1989. J.E. Flaherty (editor).

[25] R.E. Ewing, R.D. Lazarov, T.F. Russel, and P.S. Vassilevski. Local refinement via domain decomposition techniques for mixed finite element methods with rectangular Raviart-Thomas elements. In Tony Chan, editor, *Third International Symposium on Domain Decomposition for Partial Differential Equations*. SIAM, 1989.

[26] R.E. Ewing and J. Wang. Analysis of mixed finite element methods in locally refined grids. *Numer. Math.*, 63:183–194, 1992.

[27] Richard Ewing, Raytcho Lazarov, and Panayot Vassilevski. Local refinement techniques for elliptic problems on cell-centered grids I. Error analysis. *Math. Comp.*, 56(194):437–461, 1991.

[28] Richard Ewing, Raytcho Lazarov, and Panayot Vassilevski. Local refinement techniques for elliptic problems on cell-centered grids.III. algebraic multilevel BEPS preconditioners. *Numer. Math.*, 59:431–452, 1991.

[29] Richard Ewing, Raytcho Lazarov, and Panayot Vassilevski. Local refinement techniques for elliptic problems on cell-centered grids.II. Two-grids iterative methods. *Numer. Lin. Alg. with Appl*, 1(1):337–368, 1994.

[30] M. G. Fishler. Syncytial heterogeneity as a mechanism underlying cardiac far-field stimulation during defibrillation–level shock. *J. Cardiovasc. Electrophys.*, 9:384–394, 1998.

[31] David B. Geselowitz and Walter T. III Miller. A bidomain model for anisotropic cardiac muscle. *Ann. Biomed. Eng.*, 11:191–208, 1983.

[32] E. Hairer and G. Wanner. *Solving Ordinary Differential Equations II Siff and Differential Algebraic Equations*. Springer-Verlag, 1991.

[33] R.D. Hornung and J.A. Trangenstein. Adaptive mesh refinement and multilevel iteration for flow in porous media. *J. Comp. Phys.*, 136:522–545, 1997.

[34] J.P. Keener. An eikonal-curvature equation for action potential propagation in myocardium. *J. Math. Biology*, 29:629–651, 1991.

[35] J.P. Keener and K. Bogar. A numerical method for the solution of the bidomain equations in cardiac tissue. *Chaos*, 8:234–241, 1998.

[36] Wanda Krassowska and Meenakshi S. Kumar. The role of spatial interactions in creating the dispersion of transmembrane

potential by premature electric shocks. *Ann. Biomed. Eng.*, 25:949–963, 1997.

[37] Ching-Hsing Luo and Yoram Rudy. A model of the ventricular cardiac action potential: Depolarization, repolarization, and their interaction. *Circ. Res.*, 68:1501–1526, 1991.

[38] G. I. Marchuk. Splitting and alternating direction methods. In P. G. Ciarlet and J. L. Lions, editors, *Handbook of Numerical Analysis*, volume 1. North-Holland, Amsterdam, 1990.

[39] Peter K. Moore. Finite difference methods and spatial a posteriori error estimates for solving parabolic equations in three space dimensions on grids with irregular nodes. *SIAM J. Numer. Anal.*, 36:1044–1064, 1999.

[40] Adam L. Muzikant and Craig S. Henriquez. Paced activation mapping reveals organization of myocardial fibers. *J. Cardiovasc. Electrophys.*, 8:281–294, 1997.

[41] Jonathan C. Newton, Stephen B. Knisley, Xiaohong Zhou, Andrew E. Pollard, and Raymond E. Ideker. Review of mechanisms by which electrical stimulation alters the transmembrane potential. *J. Cardiovasc. Electrophys.*, 10:234–243, 1999.

[42] H. Olsson. *Runge-Kutta SOlution of Initial Value Problems.* PhD thesis, Department of Computer Science, Lund Institute of Technology, Lund University, 1999.

[43] Anand L. Pardhanani and Graham F. Carey. Efficient simulation of complex patterns in reaction-diffusion patterns. *J Comput Applied Math*, 74:295–311, 1996.

[44] Andrew E. Pollard, Nigel Hooke, and Craig S. Henriquez. Cardiac propagation simulation. *Crit Reviews Biomed Eng*, 20:171–210, 1992.

[45] John B. Pormann. *A modular simulation system for the bidomain equations.* PhD thesis, Dept. of Electrical and Computer Engineering, Duke University, 1999.

[46] Weilun Quan, Steven J. Evans, and Harold M. Hastings. Efficient integration of a realistic two-dimensional cardiac tissue model by domain decomposition. *IEEE Trans. Biomed. Eng.*, 45:372–385, 1998.

[47] Jack M. Rogers and Andrew D. McCulloch. A collocation-galerkin finite element model of cardiac action potential propagation. *IEEE Trans. Biomed. Eng.*, 41:743–757, 1994.

[48] Bradley J. Roth. Mechanisms for electrical stimulation of excitable tissue. *Crit Reviews Biomed Eng*, 22:253–305, 1994.

[49] Bradley J. Roth. A mathematical model of make and break electrical stimulation of cardiac tissue by a unipolar anode or cathode. *—TBME—*, 42:1174–1184, 1995.

[50] Bradley J. Roth and Wanda Krassowska. The induction of reentry in cardiac tissue. The missing link: How electric fields alter transmembrane potential. *Chaos*, 8:204–220, 1998.

[51] Ulrich Rüde. *Mathematical and Computational Techniques for Multilevel Adaptive Methods.* SIAM, 1993.

[52] Hasan I. Saleheen and Kwong T. Ng. A new three-dimensional finite-difference bidomain formulation for inhomogeneous anisotropic cardiac tissues. *IEEE Trans. Biomed. Eng.*, 45:15–24, 1998.

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

[54] Joakim Sudnes, Glenn Terje Lines, and Aslak Tveito. Efficient solution of ordinary differential equations modeling electrical activity in cardiac cells. Technical report, Department of Informatics, University of Oslo, 2000.

[55] Michael Taylor. *Partial Differential Equations III.* Applied Mathematical Sciences No. 117. Springer-Verlag, 1996.

[56] Vidar Thomée. *Galerkin Finite Element Methods for Parabolic Problems.* Springer-Verlag, 1997.

[57] J.A. Trangenstein. Adaptive mesh refinement for wave propagation in nonlinear solids. *SIAM J. Sci. Stat. Comput.*, 16:819–839, 1995.

[58] John A. Trangenstein and Kirill Skouibine. Operator splitting and adaptive mesh refinement for the fitzhugh-nagumo problem. Technical report, Department of Mathematics, Duke University, 2000.

[59] Natalia Trayanova and Bradley J. Roth. Cardiac tissue in an electric field: A study in electrical stimulation. *Computers in Cardiology*, 31 (Suppl.):695–698, 1992.

[60] Natalia Trayanova and Kirill Skouibine. Modeling defibrillation. effects of fiber curvature. *J. Electrocardiol.*, 31 (Suppl.):23–29, 1998.

[61] L. Tung. *A bi-domain model for describing ischemic myocardial D-C potentials.* PhD thesis, MIT, Cambridge, MA, 1978.

[62] Silvia Veronese and Hans Othmer. A computational study of wave propagation in a model for anisotropic cardiac ventricular tissue. *Lct. Notes Comput. Sc.*, 919:248–253, 1995.

[63] Edward J. Vigmond and L. Joshua Leon. Computationally efficient model for simulating electrical activity in cardiac tissue with fiber rotation. *Ann. Biomed. Eng.*, 27:160–170, 1999.

[64] James Bradley White, Gregory P. Walcott, Andrew E. Pollard, and Raymond E. Ideker. Myocardial discontinuities — a substrate for producing virtual electrodes that directly excite the myocardium by shocks. *Circulation*, 97:1738–1745, 1998.

[65] Arthur T. Winfree. Electrical instability in cardiac muscle: Phase singularities and rotors. *J. Theor. Biol.*, 138:353–405, 1989.