Operator Splitting and Adaptive Mesh Refinement for the FitzHugh-Nagumo Problem

John A. Trangenstein *Department of Mathematics, Duke University
Kirill Skouibine †Department of Mathematics, Duke University
William K. Allard ‡Department of Mathematics, Duke University

May 19, 2000

Abstract

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, [38, 63], development of spiral waves [54, 66], and effects of strong electric shocks such as used in electrical termination of arrhythmias (defibrillation) [21, 29, 40, 52, 61, 65]. 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 [46, 52, 54, 60]. A model that can account for these effects, with a proper choice of parameters, is the bidomain model [31, 62], a system of reaction-diffusion partial differential equations (PDE's) coupled with a stiff ordinary differential equation (ODE) system representing cell membrane dynamics. The bi-domain model consists of the equations for the intra- and extracellular potentials, $\Phi_i$ and $\Phi_e$, coupled through the transmembrane potential, $V_m \equiv \Phi_i - \Phi_e$:

\begin{align}
\nabla \cdot (\sigma_i \nabla \Phi_i) &= \beta I_m, \quad (1a) \\
\nabla \cdot (\sigma_e \nabla \Phi_e) &= -\beta I_m, \quad (1b) \\
I_m &= C_m \frac{\partial V_m}{\partial t} + I_{ion}, \quad (1c)
\end{align}

where $\sigma_i$ and $\sigma_e$ are intra- and extracellular conductivity tensors, $\beta$ 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 $C_m \frac{\partial V_m}{\partial t}$ 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 [6, 20, 41] (active model). Equations (1) are supplemented by boundary conditions appropriate for the problem under consideration.

*This work supported by NSF grant DMS-9870384
†This work supported by NSF grant DMS-97-5508
‡This work supported by NSF grant DMS-9870384
1.2 Survey of Numerical Methods

The complexity of the system of differential equations makes standard computer simulations expensive, thus lowering the value of the model. The objective is, therefore, to increase computational efficiency by employing advanced numerical methods.

The numerical methods previously used in simulations of cardiac electrical activity could 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 [56, 63] and ordinary workstations [5, 45, 53]; they use both explicit [16, 45, 53, 56] and semi- or fully implicit time-stepping techniques [5, 39, 51, 63, 64]. While clever implementation allowed the researchers to conduct some sample three-dimensional simulations, the resulting modeling tool is very expensive: active anisotropic models that run on massively parallel machines still 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 [56]. 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, then these 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 $\sigma_1$ and $\sigma_2$ are relatively small compared to the reaction rate, these regions will have a finite 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.

Several authors have used such adaptive algorithms successfully. A domain-decomposition method combined with an alternating direction implicit (ADI) Rush-Larsen method implemented in [34, 50] 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. 

Kirill: is this algorithm 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 (1R phase II was used in the paper) can therefore be used for simulations. A similar approach, a combination of an implicit integration technique with multigrid is employed in [47]. A large modular code has recently been developed to accommodate a wide variety of existing cardiac models and numerical approaches in [49]. 

Kirill: is this code 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 [48]. Here, it was combined with explicit, semi-implicit and fully implicit methods, as well as 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 have a factor of 5 reduction in computational time and memory expense [15]. Kirill: relative to the non-adaptive explicit algorithm? What was the max speedup possible?

Asymptotically exact 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 [43]. Although not yet applied directly to reaction-diffusion systems the results show advantage of the the spatially adaptive approach.

1.3 FitzHugh-Nagumo Model

The focus of the present study is on the development of methods that are adaptive both temporally and spatially. From the beginning, our method has been designed for extension to three dimensions, distributed computing, and a wide range of physiological applications.

However, in this paper we will describe our method for a one-dimensional problem with a very simple model. These simplifications will benefit the reader when we describe the various pieces of the algorithm (operator splitting, adaptive mesh refinement and multigrid iteration). Since this is our initial (joint) contribution to the subject area, the application of the algorithm to simple dynamics is part of the natural order of development.

We will use a modification of the two-variable FitzHugh-Nagumo model. Its dimensionless form in one dimension is

\[
\frac{\partial u}{\partial t} = \frac{\partial}{\partial x} \left( D \frac{\partial u}{\partial x} \right) + F(u, v), \quad 0 < x < 1, \quad 0 < t \quad (2a)
\]

\[
\frac{\partial v}{\partial t} = G(u, v) \quad G(u, v) \equiv \frac{\alpha}{H} u - \beta v \quad (2b)
\]

\[
\frac{\partial u}{\partial x}(x, t) = 0 \quad \text{at} \quad x = 0, 1 \quad \forall t > 0 \quad (2c)
\]

\[
u(x, 0) = u_0(x), \quad v(x, 0) = v_0(x), \quad \forall 0 < x < 1 \quad (2d)
\]

Here \( u \) represents the electrical potential and \( v \) is the gating variable. We assume that the non-dimensionalization took place in the following way. First, the space coordinate \( x \) was replaced by the dimensionless value \( x/L \), where \( L \) is the problem length. If \( u_m \) is the largest zero of the forcing function \( F \) when the gating variable is zero, then we assume that \( u \) has been replaced by \( u/u_m \). Time \( t \) has been replaced by \( tu_m \sqrt{DH}/L \); this will imply after rescaling that \( DH = 1 \). The non-dimensionalization of \( v \) is obvious after these choices. Note that a wave moving with speed \( c \) in the dimensionless system moves with speed \( cu_m \sqrt{DH} \) in the original system.

We shall make several additional assumptions on the model parameters, all of which are summarized here. The parameters \( H, \theta, \alpha \) and \( \beta \) are assumed to be nonnegative. We assume that the threshold parameter satisfies

\[
0 < \theta < \frac{1}{2} \quad (3)
\]

The fact that \( 0 < \theta < 1 \) forces \( F(u, 0) \) to have a local minimum and a local maximum in \( (0, 1) \), and the additional assumption that \( \theta < \frac{1}{2} \) guarantees that

\[
\int_0^1 F(u, 0) du = (1 - 2\theta)/12 > 0 \quad (4)
\]
so the reaction fronts have positive speed. We assume that

$$H >> 1$$ \hfill (5)

so that the forcing of the potential is stiff. We assume that

$$\alpha < 4\beta(1 - \theta)^2,$$ \hfill (6)

so that the curves \(G(u, v) = 0\) and \(F(u, v) = 0\) have the unique intersection point \((u, v) = (0, 0)\). We assume that

$$\beta = O(1) \quad \text{as} \quad H \to \infty,$$ \hfill (7)

so that the reaction \(G\) driving the gating variable \(v\) is slow compared to the reaction \(F\) driving the potential \(u\). Finally, we assume that the initial data \(v(x, 0) = v_0(x)\) involves values lying between the local minimum and local maximum of \(F(u, 0)\). Specifically, we assume that there is some point \(s\) such that

$$\forall x < s \ u(x, 0) > u_*(x) \quad \text{and} \quad \forall x > s \ u(x, 0) < u_*(x),$$

where \(u_*(x)\) is the middle root of \(F(u_*, v_0(x)) = 0\).

1.4 Overview of the Paper

In section 2 we will review the asymptotic analysis of the Fitzhugh-Nagumo model. In particular, in section 2.2 we will summarize the asymptotic behavior of traveling waves, and determine an analytical formula for the speed of the wave front at large reaction rate. This expression for the front speed will be useful in determining a timestep for the adaptive mesh refinement algorithm. We will also summarize some analytical results regarding pinning of the wave front for coarse grids, and examine numerically just how much grid is needed to obtain the correct front speed for a given set of model parameters. In section 2.4 we will examine circumstances under which reaction-diffusion systems have solutions admitting at least three bounded derivatives in time. We will use this regularity in section 2.5 to show that operator splitting of the reaction and diffusion is a second-order approximation to the reaction-diffusion problem, whenever the solution is regular.

In section 3 we will develop the basic pieces of our numerical algorithm. We will describe our techniques for integrating the reactions in section 3.2 and our techniques for integrating the diffusion in section 3.3. 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. We will discuss the selection of a timestep for these schemes in section 3.3.5, and the relative efficiency of implicit and explicit time integration in section 3.3.6. In particular, we will see that under reasonable assumptions it is necessary for computational efficiency that an iterative method for solving the linear system arising from the diffusion equation must converge in fewer than \(O(1/\Delta x)\) iterations. This will motivate the multigrid iteration, described in section 3.4.

These computational techniques for uniform grids will give us the basic tools for developing our adaptive mesh refinement scheme in section 4. We summarize the basic principles of adaptive mesh refinement in section 4.1. The adaptive grid will consist of a hierarchy of arrays of logically rectangular arrays of grid cells. Afterward, we describe modifications to timestep selection in section 4.2, and the basic recursive time-stepping strategy for adaptive mesh refinement in section 4.3.
describe how the adaptive mesh refinement algorithm choose to relocate the finer mesh in section 4.4. In particular, we describe the error estimation procedure underlying this regridding process in section 4.4.2. Techniques for communicating between scales in the mesh hierarchy are described in section 4.5; this section also discusses some techniques to avoid taking fine timesteps on coarse grid. We describe modifications to the iterative linear algebra for adaptive mesh refinement in section 5. 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 5.1. The iterative scheme involves conjugate gradients, preconditioned by a multiplicative domain decomposition. The multiplicative domain decomposition algorithm is described in section 5.3. In order to take advantage of the organization of the grid hierarchy, in section 5.4 we will describe a multiplicative domain decomposition smoother that works well on an array of grid patches, and extends well to distributed computing.

Finally, we will present some numerical results in section 6.

2 Analysis of the Fitzhugh-Nagumo Model

2.1 Linearized Stability in the Absence of Diffusion

The dynamics of the reaction-diffusion system (2) are well-understood in the absence of diffusion [30, 44]. When $D = 0$, the stationary states of the reaction satisfy $\frac{du}{dt} = v = Hu(u - \theta)(1 - u)$. Thus, at a stationary state either $u = 0$ or $\frac{du}{dt} = (u - \theta)(1 - u) = -u^2 + (1 + \theta)u - \theta$. Since assumption (6) guarantees that this quadratic has no real solutions, the only stationary state of the reaction is $(u, v) = (0, 0)$.

The dynamics of this system for large values of $u$ and $v$ are easy to understand. Looking at Figure 1, it is easy to see that at every point around the boundary of the figure the trajectory of the Fitzhugh-Nagumo reaction moves into the interior of the figure. As a result, the orbits are bounded.

![Figure 1: Forcing functions in Fitzhugh-Nagumo Model](image)

A linearized stability analysis can be used to determine the local stability of the stationary point.
The matrix of partial derivatives of the forcing functions $F$ and $G$ is

$$
\begin{bmatrix}
\frac{\partial F}{\partial \tau}(0, 0) & -1 \\
H \alpha & -\beta
\end{bmatrix} = \begin{bmatrix}
-H \theta & -1 \\
H \alpha & -\beta
\end{bmatrix}.
$$

Since the trace of this matrix is negative and the determinant is positive, there are two negative eigenvalues. As a result, the origin is a linearly stable stationary point.

### 2.2 Traveling Waves

It is well-known [33, 36, 44] that the Fitzhugh-Nagumo model exhibits traveling waves for certain choices of the model parameters and certain initial data. We will summarize the asymptotic analyses by these authors in this section, since we could not find a single source that contained all this information.

Recall assumption (5) that $H >> 1$ and the result from rescaling time that $DH = 1$. In order to follow the discussion in [36] we will define

$$
\epsilon = \frac{1}{H}, \quad w = \frac{v}{H},
$$

and

$$
f(u, w) \equiv F(u, v)/H = u(u - 6)(1 - u) - w, \quad g(u, w) \equiv G(u, v)/H = \alpha u - \beta w. \quad (8)
$$

We can now rewrite the Fitzhugh-Nagumo model in the form

\begin{align}
\frac{\partial u}{\partial t} &= \epsilon \frac{\partial^2 u}{\partial x^2} + \frac{1}{\epsilon} f(u, w), \quad 0 < x < 1 \quad 0 < t \quad (9a) \\
\frac{\partial w}{\partial t} &= g(u, w), \quad 0 < x < 1 \quad 0 < t. \quad (9b)
\end{align}

Equation (8) shows that $f(u, w) = 0$ can easily be solved for $w$; the resulting function $w(u)$ has a local minimum for some $u \in (0, \theta)$ and a local maximum for some $u \in (\theta, 1)$. For $w$ greater than the local minimum, we can solve $f(u, w) = 0$ for $u$ to obtain the smallest possible solution $u = u_-(w)$. Similarly, for $w$ less than the local maximum we can solve $f(u, w) = 0$ for $u$ to obtain largest possible solution $u = u_+(w)$. Between the local minimum and local maximum of $w(u)$ we can solve $f(u, w) = 0$ to get the middle solution $u = u_*(w)$.

### 2.2.1 Asymptotics at Early Time $t \leq O(\epsilon)$

For small times, we can let

$$
\tau = t/\epsilon, \quad u(x, t) = U(x, \tau) \text{ and } w(x, t) = W(x, \tau). \quad (10)
$$

Then the Fitzhugh-Nagumo equations (9) can be rewritten

$$
\frac{\partial U}{\partial \tau} = \epsilon^2 \frac{\partial^2 U}{\partial x^2} + f(U, W), \quad \frac{\partial W}{\partial \tau} = \epsilon g(U, W).
$$

Suppose that we have asymptotic expansions of the form

$$
U(x, \tau; \epsilon) = U_0(x, \tau) + \epsilon U_1(x, \tau) + \ldots, \quad W(x, \tau; \epsilon) = W_0(x, \tau) + \epsilon W_1(x, \tau) + \ldots.
$$
Then by separating terms in the differential equations by powers of $\epsilon$ we obtain
\[
\frac{\partial U_0}{\partial \tau} = f(U_0, W_0), \quad \frac{\partial W_0}{\partial \tau} = 0,
\]
and so on. The leading terms in the asymptotic expansion indicates that at all points in space, $W_0$ remains fixed at its initial value. The differential equation for $U_0$ is thus an ordinary differential equation in time, parameterized by spatial location. At a given spatial location $x$, if the initial data satisfies $u(x, 0) < u_*(w(x, 0))$ then for large $\tau$ (i.e., $t > O(\epsilon)$; $U_0(x, t)$ will approach $u_-(w(x, 0))$. Similarly, if the initial data satisfies $u(x, 0) > u_*(w(x, 0))$, then $U_0(x, t)$ will approach $u_+(w(x, 0))$. In other words, $u(x, t)$ will tend rapidly to either $u_-(w(x, 0))$ or $u_+(w(x, 0))$.

2.2.2 Asymptotics at Time $t = O(1)$

Suppose that the initial data is such that at some point $s$ we have a discontinuity:
\[
u(x, 0) = \begin{cases}
u_+(w(x, 0)) + O(\epsilon), & x < s \\ (10) \\
u_-(w(x, 0)) + O(\epsilon), & x > s
\end{cases}
\]

These initial data correspond to using the leading order results from section 2.2.1 for the early time evolution of the system with initial data such that $u(x, 0) > u_*(w(x, 0))$ for $x < s$, and $u(x, 0) < u_*(w(x, 0))$ for $x > s$.

We shall assume that the initial discontinuity evolves into a front that moves with speed $s(t)$. We will transform into a frame of reference moving with this front by taking
\[
\xi = x - s(t), \quad u(x, t) = U(\xi, t) \text{ and } w(x, t) = W(\xi, t).
\]

(Note that $U$ and $W$ are different functions from those in section 2.2.1; we distinguish these functions solely by their arguments.) These definitions imply that
\[
\frac{\partial U}{\partial t} = \frac{\partial U}{\partial \xi} s'(t) + \epsilon \frac{\partial^2 U}{\partial x^2} + \frac{1}{\epsilon} f(U, W), \quad \frac{\partial W}{\partial t} = \frac{\partial W}{\partial \xi} s'(t) + g(U, W).
\]

If we assume asymptotic expansions of the form
\[
U(\xi, t; \epsilon) = U_0(\xi, t) + \epsilon U_1(\xi, t) + \ldots, \quad W(\xi, t; \epsilon) = W_0(\xi, t) + \epsilon W_1(\xi, t) + \ldots
\]
then separating terms in the differential equations by powers of $\epsilon$ leads to
\[
f(U_0, W_0) = 0, \quad \frac{\partial W_0}{\partial t} - \frac{\partial W_0}{\partial \xi} \frac{\partial s}{\partial t} = g(U_0, W_0),
\]
and so on. Since the former of the first-order equations shows that $U_0$ is constrained by the nonlinear function $f$, we must have
\[
U_0(\xi, t) = \begin{cases}
u_+(W_0(\xi, t)), & \xi < 0 \\
u_-(W_0(\xi, t)), & \xi > 0
\end{cases}
\]
Using this result for $U_0$, we can write the equation for $W_0$ in the form of an ordinary differential equation (parameterized by $\xi$)
\[
\frac{dW_0}{dt} = \begin{cases}
g(u_+(W_0), W_0), & \xi < 0 \\
g(u_-(W_0), W_0), & \xi > 0
\end{cases}
\]
(12)

This is an ordinary differential equation in $t$, parameterized by $\xi$. If we knew the evolution of the discontinuity $s(t)$ we could use the solution of these ordinary differential equations to determine the leading term in the asymptotic expansion for $u$. 

2.2.3 Internal Structure of the Wave Front

In order to determine the speed of the wave front, we will define

\[ \eta = (x - s(t))/\epsilon, \quad u(x, t) = U(\eta, t) \text{ and } w(x, t) = W(\eta, t). \]

The application of this change of variables in (9) implies that

\[ \frac{\partial U}{\partial t} = \frac{\partial U}{\partial \eta} \frac{\partial s}{\partial \eta} + \frac{1}{\epsilon} \frac{\partial^2 U}{\partial \eta^2} + \frac{1}{\epsilon} f(U, W), \quad \frac{\partial W}{\partial t} = \frac{\partial W}{\partial \eta} \frac{\partial s}{\partial \eta} + g(U, W), \]

If we assume asymptotic expansions of the form

\[ U(\eta, t; \epsilon) = U_0(\eta, t) + \epsilon U_1(\eta, t) + \ldots, \quad W(\eta, t; \epsilon) = W_0(\eta, t) + \epsilon W_1(\eta, t) + \ldots \]

then by separating terms in the differential equations by powers of \( \epsilon \) we obtain

\[ s' \frac{\partial U_0}{\partial \eta} + \frac{\partial^2 U_0}{\partial \eta^2} + f(U_0, W_0) = 0, \quad \frac{\partial W_0}{\partial \eta} s' = 0, \tag{13} \]

and so on. If the front is not stationary, then the latter of these equations shows that \( W_0 \) is independent of \( \eta \); in other words, \( W_0 \) is a function of \( t \) only. The former of the two equations is a boundary value problem for \( U_0 \), parameterized in time by \( W_0(t) \) and \( s' \), in which

\[ U_0(\eta, t) \rightarrow \begin{cases} u_+(W_0(t)), & \eta \to -\infty \\ u_-(W_0(t)), & \eta \to \infty \end{cases}. \tag{14} \]

2.2.4 Determination of the Front Speed

As suggested by the results of section 2.2.3, suppose that at some fixed time \( t \) we are given a fixed \( W_0 \) between the local min and local max of \( f(u, 0) \). Recall assumption (4), which implies that

\[ \int_{u_-(W_0)}^{u_+(W_0)} f(u, W_0) \, du > 0. \]

Let

\[ \tilde{u}(\eta) = U_0(-\eta, t), \quad \tilde{f}(\tilde{u}) = f(\tilde{u}, W_0), \quad \tilde{u}_- = u_-(W_0), \quad \tilde{u}_+ = u_+(W_0). \]

We want to find \( c = s'(t) \) so that we can solve the boundary-value problem defined in (13) and (14). In terms of \( \tilde{u} \), this can be written

\[ \tilde{u}'' - c\tilde{u}' + \tilde{f}(\tilde{u}) = 0, \quad \lim_{\eta \to -\infty} \tilde{u}(\eta) = \tilde{u}_-, \quad \lim_{\eta \to \infty} \tilde{u}(\eta) = \tilde{u}_+. \tag{15} \]

Following [33], we could use arguments from dynamical systems to show that there is a unique front speed \( c \) that solves the two-point boundary-value problem (15). This leads to the interesting formula

\[ c = \frac{\int_{\tilde{u}_-}^{\tilde{u}_+} \tilde{f}(\zeta) \, d\zeta}{\int_{\tilde{u}_-}^{\infty} (\tilde{u}'(\eta))^2 \, d\eta}. \]

However, this does not give us a formula for the front speed that is useful in simulation.
However, it is possible to find the analytical solution for $\tilde{u}$. Suppose that we substitute

$$y(\eta) = \frac{1}{1 + e^{-\lambda \eta}}, \quad \tilde{u}(\eta) = \chi(y),$$

where $\lambda > 0$ is to be determined. Then $\frac{dy}{d\eta} = \lambda y(1 - y)$, and $\chi(y)$ satisfies

$$\frac{d^2 \chi}{dy^2} \lambda^2 y^2 (1 - y)^2 + \frac{d \chi}{dy} y(1 - y)[-c \lambda + \lambda^2(1 - 2y)] + (\chi - \tilde{u}_-)(\chi - \tilde{u}_+)(\tilde{u}_+ - \chi) = 0.$$

$$\chi(0) = \tilde{u}_-, \quad \chi(1) = \tilde{u}_+.$$

Next, if we substitute

$$\chi(y) = \tilde{u}_- + (\tilde{u}_+ - \tilde{u}_-)y$$

into the differential equation, we see that $\chi(y)$ solves this problem provided that $c$ and $\lambda$ are chosen so that

$$0 = (\tilde{u}_+ - \tilde{u}_-) \lambda^2 y(1 - 2y)(1 - y) - c(\tilde{u}_+ - \tilde{u}_-) \lambda y(1 - y),$$

and

$$-h(\tilde{u}_+ - \tilde{u}_-)(y - 1)[\tilde{u}_- - \tilde{u}_+ + y(\tilde{u}_+ - \tilde{u}_-)y(\tilde{u}_+ - \tilde{u}_-)].$$

If we cancel out the common factors, we are left with a linear function of $y$ that is identically zero. The coefficients in this linear function are

$$-2\lambda^2 + (\tilde{u}_+ - \tilde{u}_-) = 0, \quad \lambda^2 - c\lambda - (\tilde{u}_+ - \tilde{u}_-)(\tilde{u}_+ - \tilde{u}_-) = 0.$$

These equations imply that

$$\lambda = (\tilde{u}_+ - \tilde{u}_-)/\sqrt{2}, \quad c = \sqrt{2}(\tilde{u}_+ + \tilde{u}_-)/2 - \tilde{u}_-.$$

In terms of the original variables, this implies that the speed of the front is $\sqrt{\tilde{H}/H}(\tilde{u}_+ + \tilde{u}_-)/2 - \tilde{u}_-.$

### 2.2.5 Asymptotics Behind the Front

Recall from the asymptotic analysis in section 2.2.2 that we developed equations (12) for $U_c(\xi, t)$ and $W_0(\xi, t).$ With our assumed initial conditions (10) the equation for $U_0$ implies that

$$U_c(\xi, t) = \begin{cases} u_-(W_0(\xi, t)), & \xi > 0 \\ u_+(W_0(\xi, t)), & \xi < 0 \end{cases}$$

The equation for $W_0$ can be written

$$\frac{dW_0}{dt} = g(U_0, W_0) = \begin{cases} g(u_-(W_0(\xi, t)), W_0(\xi, t)), & \xi > 0 \\ g(u_+(W_0(\xi, t)), W_0(\xi, t)), & \xi < 0 \end{cases}$$

Together with the initial data $W_0(\xi, 0) = w(\xi, 0) = v(\xi, 0)/H,$ this gives us ordinary differential equations for $W_c$ for each spatial location $\xi.$
Note that these differential equations do not depend on knowledge of the front location \( s(t) \) or the speed of the front. Thus, after we solve these ordinary differential equations for \( W_0 \), we can compute the speed of the front as a function of time, and integrate to determine the position of the front. Afterwards, we can perform the coordinate transformation from \( \xi \) back to \( x \).

On either side of the front, \( W_0 \) will tend to a stationary point. Behind the front, the stationary point is \( W_0 = w_+ \), where
\[
g(u_+(w_+), w_+) = 0.
\]

Ahead of the front, there is a similarly-determined stationary point for \( W_0 \). It follows that at large time \( t > O(1) \) the speed of the front will tend to the speed associated with these two stationary values. This gives us the information we need to estimate the front speed for numerical computations.

Much more can be said about the asymptotic analysis of the the Fitzhugh-Nagumo system. For additional information, the reader should consult \([30, 44]\). In particular, there are interesting questions regarding those conditions that cause the model to be excitable. In addition, we note that there is typically a second front behind the initial front, with similar dynamics. It is interesting that this second front represents one of the limitations of the Fitzhugh-Nagumo model: the observed behavior in electrical wave propagation in the heart is that the behavior is smooth after the first front passes by.

### 2.2.6 Remarks on the Computation of the Front Speed

The analysis we have presented here is instructive, but not necessarily quantitative. There are restrictive assumptions regarding the relative size of reaction and diffusion, and formulas that are difficult to use inside computations. There is a separate discussion, which can be found in \([36, 44]\) that the form of the initial data may affect the speed of the front. This analysis was performed on unbounded domains, and requires somewhat special numerical treatment to reproduce \([42]\).

In practice, it is common to know the speed of the electrical wave front from experiments. The information regarding the reaction rates is known from experiments on individual heart cells. It is difficult to measure the diffusion, because this depends on the behavior of a region of tissue, consisting of a large number of cells. It is common to determine the diffusion through numerical experiments, by adjusting the diffusion until the physical front speed is obtained.

Using the algorithm we will describe later in this paper for a uniform grid, we have investigated the numerical convergence of the front speed as the mesh and timestep size are refined. In order to compute the front speed at some timestep, we searched the numerical results to find two successive points where \( F(u, v) \geq 0 \) on the left and \( F(u, v) < 0 \) on the right. We used linear interpolation to find the value of the potential \( u \) where \( F = 0 \), and linear interpolation to find the location in space where this value of \( u \) would occur. Then we computed the front speed by computing the difference between the current spatial location of the front and the previous value, divided the difference in times. The previous value of the front position was determined at some arbitrary time after startup, in this case \( t = 0.2 \).

The results in Figure 2 show how the front speed depends on the reaction rate \( H \) and the threshold \( \theta \). The results show that the front speed approaches the value predicted in section 2.2.4 as the reaction rate \( H \) becomes large. These same numerical results indicate that the front width is inversely proportional to \( H \) (\( \approx 8.3/H \) at \( \theta = 0.25 \)) and independent of \( \theta \); this is consistent with the asymptotic analysis in section 2.2.3. Here, the width of the front was computed between the points where the potential achieved 95% and 5% of its values on either side of the front. We also see that the front speed is proportional to \( 1/\theta \approx 1.4(1/\theta + \theta) \) at \( H = 100 \). Note that when we changed \( H \) we also changed \( D = 1/H \); when we changed \( \theta \) we kept \( H \) and \( D \) fixed.
The results in Figure 3 show that the computed front speed depends on the mesh width. As the strength $H$ of the reaction for the electrical potential is increased, the number of cells required to obtain the correct front speed also increases. In order to obtain a front speed accurate to roughly 2 digits, it appears to be necessary to refine the mesh so that there are on the order of 30 grid cells in the reaction front. Note that the front speed increases monotonically from zero speed (pinning, discussed in the next section 2.3), to the limiting speed as the mesh is refined. These results are similar to the results reported in [51].

![Graphs](image)

**Figure 2**: Front speed versus mesh model parameters. (a) Front speed versus reaction rate $H$ for fixed threshold $\theta = 0.25$; (b) front speed versus $\theta$ for fixed $H = 100$. The analytical value of the front speed at large $H$ is $\sqrt{2}/4 \approx 0.35$.

### 2.3 Pinning

Although the analysis of the front speed for the FitzHugh-Nagumo model shows that there are propagating wave fronts whenever $\theta \neq \frac{1}{2}$, numerical methods for solving this problem do not always produce discrete traveling wave fronts. It is known that for coarse grids, the numerical solutions may fail to propagate on coarse grids. This naturally leads us to ask how fine a mesh is necessary, both to produce a propagating numerical wave, and to produce the correct speed of the numerical wave.

These questions have been addressed, to some extent, by Keener [37]. He considered the semi-discrete system of ordinary differential equations

$$\frac{du_j}{dt} = \frac{D}{\Delta x^2} [u_{j+1} - 2u_j + u_{j-1}] + F(u_j; 0), -\infty < j < \infty, 0 < t.$$

Here the forcing function $F$ is defined by (2a).

Keener showed that if

$$\lambda \lambda x > \frac{\sqrt{8}}{H \min \{\theta, 1 - \theta\}},$$

then the semi-discrete system cannot produce propagating waves. (Recall that we are assuming that the problem has been non-dimensionalized so that $\sqrt{DH} = 1$.) Notice that the most severe restriction
Figure 3: Front speed versus mesh width. The analytical value of the front speed at large reaction rate $H$ is $\sqrt{2}/4 \approx 0.35$. (a) Results for $H = 100$; (b) Results for $H = 1000$.

on $\Delta x$ occurs when $\theta = \frac{1}{2}$, the point at which the analytical solution does not have traveling waves. In addition, he showed that for $0 \leq \theta < \frac{1}{2}$ and

$$\Delta x < \frac{2}{H\theta}$$

(16)

the semi-discrete system has propagating wave solutions. In addition, he showed that the relative error in the front speed for the semi-discrete system as an approximation to the continuous front speed is $O(\Delta x^2)$.

Of course, we are interested in fully discrete numerical methods for solving problem (2). Using the algorithm described below for a uniform grid, we have experimentally determined the smallest number of cells that will allow the wave front to propagate. The results are shown in Figure 4. The results in Figure 4a show that the minimum number of cells that allow the front to move is linearly dependent on the reaction rate $H$ in (2a). Figure 4b shows that for $\theta \to 0$, $\Delta x$ is inversely proportional to $\theta$, as suggested by Keener [37]. Figure 4b also shows that as $\theta \to \frac{1}{2}$ we find that $\Delta x \to 0$; this is similar to the results in [7].

2.4 Regularity

Numerical methods for solving partial differential equations typically assume some order of smoothness on the solutions in order to obtain error estimates. In this section, we will show that the solution of the Fitzhugh-Nagumo equations (2) is smooth, provided that the initial data for the gating variable is continuous. The proof will involve showing that the Fitzhugh-Nagumo problem involves invariant regions in the potential and gating variable. Our development is different from that in [57], who proves well-posedness for reaction-diffusion systems and the existence of invariant regions.

2.4.1 Preliminaries

Taylor’s Theorem
Figure 4: Minimum number of cells to avoid pinning. (a) Number cells versus reaction rate for fixed $\theta = 1/4$. The number of cells is a linear function of the reaction rate; in this case $H\Delta x \approx 51$. (b) Number cells versus threshold for fixed $H = 1000$. The number of cells becomes small as the threshold tends either zero or to one-half.

**Lemma 2-1:** Let $J$ be a positive integer. Suppose $g$ is a $J$ times continuously differentiable function with values in a finite dimensional vector space defined on an open subset of $\mathbf{R}$ containing $[0, 1]$. Then,

$$
g(1) = \sum_{0 \leq j < J} \frac{1}{j!} g^{(j)}(0) - \int_{0}^{1} g^{(J)}(\tau) \frac{(1 - \tau)^{J}}{J!} \, d\tau.
$$

(17)

**Proof:** This is a straightforward application of integration by parts and induction. $\square$

**Lemma 2-2:** Suppose $V$ and $W$ are finite dimensional vector spaces. For each nonnegative integer $j$ let $\mathbf{P}^j(V, W)$ be the vector space of $W$-valued homogeneous polynomial functions on $V$ of degree $j$. Suppose $G$ is an open subset of $V$, $J$ is a positive integer and $f$ is $J$ times continuously differentiable function on $G$ with values in $W$. For each nonnegative integer $j$ not exceeding $J$ we define the function

$$
f^{(j)} : G \rightarrow \mathbf{P}^j(V, W)
$$

at $a \in G$ by setting

$$
f^{(j)}(a)(u) = \left. \frac{d^j}{dt^j} f(a + tu) \right|_{t=0} \quad \text{for} \quad u \in V.
$$

Let $H = \{(a, x) \in G \times G : \{(1 - t)a + tx : 0 \leq t \leq 1\} \subset G\}$ and note that $H$ is an open subset of $G \times G$. We define the function

$$
f^{(J)} : H \rightarrow \mathbf{P}^j(V, W)
$$
at \((a, x) \in H\) by setting
\[
 f^{(J)}(a, x)(u) = -\int_0^1 f^{(J)}((1-t)a + tx;)(u)(1-t)^J \quad \text{for} \ u \in U.
\]

Then
\[
 \forall (x, a) \in H, \ f(x) = \sum_{0 \leq j < J} \frac{1}{j!} f^{(j)}(a)(x-a) + \frac{1}{J!} f^{(J)}(x, a)(x-a)
\]

**Proof:** This follows by letting \(g(t) = f(a + t(x-a))\) in (17) above. \(\Box\)

**The spaces**

We let \(H_0\) be the pseudo-Hilbert space of real valued square summable functions on \(\mathbb{R}\) which are 2-periodic and even. We set
\[
 (u, v) = \int_0^1 u(x)v(x) \, dx \quad \text{and} \quad ||u|| = \sqrt{(u, u)} \quad \text{for} \ u, v \in H_0.
\]

For each nonnegative integer \(m\) we set
\[
 C_m(x) = \cos mx \quad \text{for} \ x \in \mathbb{R}
\]
and note that \(C_m \in H_0\). As is well known [67],
\[
 \lim_{M \to \infty} ||u - (u, C_0)C_0 - \frac{1}{2} \sum_{m=1}^M (u, C_m)C_m|| = 0 \quad \text{for each} \ u \in H_0.
\]

For each positive integer \(s\) we let \(H_s\) be the set of those \(u \in H_0\) for which
\[
 ||u||_s^2 = (u, C_0)^2 + \frac{1}{2} \sum_{m=1}^\infty m^{2s}(u, C_m)^2 < \infty.
\]

Evidently, \(H_s \subset H_t\) if \(s < t\). As is well known, \(H_s\) is a pseudo-Hilbert space with respect to the norm \(|| \cdot ||_s\). We define the linear map
\[
 I : H_2 \to H_0
\]
by requiring that \((I(u), v) = (u, v^m)\) whenever \(u \in H_2\) and \(v \in H_\infty\). It is elementary that
\[
 ||I(u)||_s^2 + (u, C_0)^2 = ||u||_{s+2}^2 \quad \text{whenever} \ u \in H_{s+2}.
\]

For each \(t > 0\) we define the linear map
\[
 K_t : H_0 \to H_\infty
\]
by setting
\[
 K_t(u) = (u, C_0)C_0 + \frac{1}{2} \sum_{m=1}^\infty e^{-m^2t}(u, C_m)C_m \quad \forall u \in H_0.
\]
Evidently,
\[ ||K_t(u)||_s \leq ||u||_s \quad \text{and} \quad \lim_{t \to 0} ||K_t(u) - u||_s = 0 \quad \forall u \in \mathbf{H}_s \]

Moreover, for any \( u \in \mathbf{H}_c \) and any nonnegative integer \( s \),

\[ 0 < t \mapsto K_t(u) \] is a smooth path in \( \mathbf{H}_s \)

whose velocity satisfies
\[ \frac{d}{dt} K_t(u) = L(K_t(u)). \]

It is evident that
\[ L(K_t(u)) = K_t(L(u)) \quad \forall u \in \mathbf{H}_2 \]

**Lemma 2-3:** For each positive integer \( J \) we set

\[ K_t^{(J)}(u) = K_t(u) - \sum_{0 \leq j < J} \frac{t^j}{j!} L^j(u) \quad \forall u \in \mathbf{H}_{2(J-1)} \]

Then
\[ ||K_t^{(J)}(u)||_s \leq \frac{t^J}{J!} ||u||_{s+2J} \] \hspace{1cm} (18)

and
\[ \lim_{t \to 0} t^{-J} ||K_t^{(J+1)}(u)||_s = 0 \] \hspace{1cm} (19)

whenever \( u \in \mathbf{H}_{s+2J} \).

**Proof:** Setting \( g(\tau) = K_{\tau t}(u) \) in (1) we find that

\[ K_t^{(J)}(u) = -t^J \int_0^1 K_{\tau t} L^J(u) d \frac{(1-\tau)^J}{J!} \]

so (18) follows from (1). Moreover,

\[ K_t^{(J+1)}(u) = -t^J \int_0^1 K_{\tau t} L^J(u) d \frac{(1-\tau)^J}{J!} - L^J(u) d \frac{(1-t)^J}{J!} \]

so (19) follows from (1). \( \square \)

**Lemma 2-4:** For any \( u \in \mathbf{H}_1 \) we have

\[ (\text{ess sup} \ |u| \leq ||u||_0 + ||u||_1 \]

and

\[ |u(x) - u(y)| \leq ||u||_1 |x - y|^{\frac{1}{2}} \quad \text{for almost all} \ x, y \in \mathbf{R} \]
Proof: Suppose $u$ is a smooth $\mathbb{R}$-valued even 2-periodic function on $\mathbb{R}$. Then

$$u(y) - u(x) = \int_x^y u'(t) \, dt \quad x, y \in [0, 1]$$

which by the Schwartz inequality implies that

$$|u(y) - u(x)| \leq ||u||_1 |x - y|^{1/2}, \quad x, y \in [0, 1].$$

In particular, $|u(y)| \leq |u(x)| + ||u||_1$: Integrating this inequality from 0 to 1 with respect to $x$ and using the Schwartz inequality again yields $|u(y)| \leq ||u||_c + ||u||_1$. The statement to be proved follows by approximation. □

We let $H_{\infty} = \bigcap_{n=1}^{\infty} H_n$. Applying the preceding Proposition repeatedly we find that each member of $H_{\infty}$ is almost equal a smooth function: as is obvious, each real valued smooth 2-periodic and even function is a member of $H_{\infty}$.

Suppose now $\mathbf{U}$ is a finite dimensional inner product space. For each $s = 0, 1, 2, \ldots$ we let $H_s(\mathbf{U})$ be the pseudo-Hilbert space of those $U : \mathbb{R} \to \mathbf{U}$ such that $U \circ u \in H_s$ whenever $u \in \mathbf{U}$ and where

$$||U||_s^2 = \sum_{j=1}^{N} ||U \circ u_j||_s^2$$

whenever $u_1, \ldots, u_N$ is an orthonormal basis for $\mathbf{U}$.

Some consideration of ordinary differential equations.

Suppose $\mathbf{U}$ is a finite dimensional vector space, $G$ is an open subset of $G$ and $h : G \to \mathbf{U}$ is smooth. (We have in mind $\mathbf{U} = \mathbb{R}^2$ and $h(U) = (F(u, v), G(u, v))$ for $U = (u, v) \in \mathbb{R}^2$ where $F, G$ are as in the Fitzugh-Nagumo equations.)

Let $s$ be the set of those $((t, U), V) \in (\mathbb{R} \times G) \times G$ such that either $t = 0$ and $U = V$ or $t \neq 0$ and there is a continuously differentiable function $\gamma$ carrying some open interval $I$ with $\{0, t\} \subset I$ into $\bar{G}$ such that

$$\frac{d}{d\tau} \gamma(\tau) = h(\gamma(\tau)) \quad \forall \tau \in I.$$  

It is well known that $s$ is smooth function whose domain is an open subset of $\mathbb{R} \times G$. Let $H$ be the domain of $s$. Evidently, $\{0\} \times G \subset H$; for any $U \in \mathbb{R}^2$, $\{t \in \mathbb{R} : (t, U) \in H\}$ is an open interval; the range of $s$ equals $G$;

$$s(0, U) = U \quad \text{for} \quad U \in G;$$

and

$$\frac{d}{dt} s(t, U) = h(s(t, U)) \quad \text{whenever} \quad (t, U) \in H.$$  

By Taylor’s Theorem, there are unique smooth functions $j$ on $G$ and $r_2, r_3$ on $H$ such that

$$s(t, U) = U + th(U) + t^2 r_2(t, U)$$
and
\[ s(t, U) = U + t h(U) + t^2 j(U) + t^3 r_3(t, U) \]
for any \( (t, U) \in H \). For \( (t, U) \in H \) we set
\[ s_t(U) = s(t, U), \quad r_{2,t}(U) = r_2(t, U), \quad r_{3,t}(U) = r_3(t, U) \]
Suppose \( U \in G \). From the chain rule applied to \( (\ref{eq:2.3}) \) we infer that
\[ \frac{d}{dt} s(t, U)|_{t=0} = h^{(1)}(U)(h(U)) ; \]
on the other hand, it is clear from \( (\ref{eq:2.4}) \) that
\[ \frac{d}{dt} s(t, U)|_{t=0} = j(U), \]
Thus
\[ j(U) = h^{(1)}(U)(h(U)) \quad \text{for} \quad U \in G. \]

2.4.2 The error estimates.

The equations.
Fix \( T \in (0, \infty) \). For \( U = (u, v) \in H_2(\mathbb{R}^2) \) we set
\[ M(U) = (L(u), 0), \quad N_t(U) = (K_t(u), v), \quad t > 0, \quad N_t^{(J)}(U) = (K^{(J)}(u), v), \quad t > 0, \quad J = 1, 2, \ldots. \]

We adopt the notation of 1.3, with \( b \) and \( U \) real \( \mathbb{R}^2 \) and \( h = (F, G) \) and we set \( S_t(U)(x) = s_t(U)(x) \), etc. We say \( U \in C^1([0, T], H_2(\mathbb{R}^2)) \) is a solution if
\[ \frac{d}{dt} U = M(U) + H(U) \quad \text{on} \quad [0, T]. \]

Note that if \( U \in C^2([0, T], H_2(\mathbb{R}^2)) \) then
\[ \frac{d}{dt}^2 U(t)|_{t=0} = (M + H^{(1)}(U(t)))(M + H)(U(t)). \]

The first order estimate.
Suppose \( U \in H_2(\mathbb{R}^2) \). We have
\[ N_t(S_t(U)) = S_t(U) + t M(S_t(U)) + N_t^{(2)}(S_t(U)) \]
\[ = U + t H(U) + \frac{t^2}{2} R_{2,t}(U) + t M(U) + t^2 M(R_{1,t}(U)) + N_t^{(2)}(S_t(U)) \]
\[ = U + t (M(U) + H(U)) + C_t(U), \]
where we have set
\[ C_t(U) = \frac{t^2}{2} R_{2,t}(U) + t^2 M(R_{1,t}(U)) + N_t^{(2)}(S_t(U)). \]
and where
\[
\lim_{t \to 0} t^{-1} \| C_t(U) \|_{L^1} = 0 \quad \forall U \in H_{s+2},
\]
\[
\lim_{t \to 0} t^{-2} \| C_t(U) \|_{L^2} = 0 \quad \forall U \in H_{s+4}.
\]
One may treat \( S_t(N_t(U)) \) in a similar fashion.

**The second order estimate.** Suppose \( U \in H_4(\mathbb{R}^2) \). We set
\[
T_t(U) = S_\frac{t}{2}(N_t(S_{\frac{t}{2}}(U))),
\]
\[
T_{0,t}(U) = N_t(S_{\frac{t}{2}}(U)),
\]
\[
T_{1,t}(U) = \frac{t}{2} H(T_{0,t}(U)),
\]
\[
T_{2,t}(U) = \frac{t^2}{4} J(T_{0,t}(U)),
\]
\[
T_{3,t}(U) = \frac{t^3}{6} R_{3,t}(T_{0,t}(U)),
\]
\[
A(U) = M(U) + \frac{1}{2} H(U),
\]
\[
B(U) = \frac{1}{2} M^2(U) + \frac{1}{4} M(H(U)) + \frac{1}{4} J(U),
\]
\[
C_{1,t}(U) = N^{[3]}_t(U) + \frac{t}{2} N^{[2]}_t(H(U)) + \frac{t^2}{4} N^{[1]}_t(H(U)) + \frac{t^3}{8} N_t(R_{3,t}(U)),
\]
\[
C_{2,t}(U) = t H^{[1]}(U)(t^2 B(U)) + C_{1,t}(U)) + \frac{t}{2} H^{[2]}(U; T_{1,t}(U), tA(U) + t^2 B(U) + C_{1,t}(U))
\]
\[
C_{3,t}(U) = \frac{t^2}{4} J^{[1]}(U; T_{0,t}(U))(tA(U) + t^2 B(U) + C_{1,t}(U))
\]

Evidently,
\[
T_t(U) = T_{0,t}(U) + T_{1,t}(U) + T_{2,t}(U) + T_{3,t}(U).
\]
We have

\[ T_{0,t}(U) = N_t(U) + \frac{t}{2} M(U) - \frac{t^2}{4} N_t^2(U) + \frac{t}{8} N_t(R_{3,t}^4(U)) \]

\[ = U + tM(U) + \frac{t^2}{2} M^2(U) + N_t^2(U) \]

\[ + \frac{t}{2} H(U) + \frac{t^2}{4} M(H(U)) + \frac{t}{2} N_t^2(H(U)) \]

\[ + \frac{t^2}{4} J(U) + \frac{t^2}{4} N_t^2(J(U)) \]

\[ + \frac{t^3}{8} N_t(R_{3,t}^4(U)) \]

\[ = U + tA(U) + t^2B(U) + C_{1,t}(U) \]

\[ T_{1,t}(U) = \frac{t}{2} H(T_{0,t}(U)) \]

\[ = \frac{t}{2} (H(U) + H^{(1)}(U)(T_{0,t}(U) - U) + \frac{1}{2} H^{(2)}(U, T_{0,t}(U))(T_{0,t}(U) - U)) \]

\[ = \frac{t}{2} H(U) + \frac{t^2}{2} H^{(1)}(U)(A(U)) + C_{2,t}(U), \]

\[ T_{2,t}(U) = \frac{t^2}{4} J(U) + \frac{t^2}{4} J^{(1)}(U, T_{0,t}(U))(T_{0,t}(U) - U) \]

\[ = \frac{t^2}{4} J(U) + C_{3,t}(U) \]

Keeping in mind that

\[ J(U) = H^{(1)}(U)(H(U)) \]

we find that

\[ T_t(U) = T_{0,t}(U) + T_{1,t}(U) + T_{2,t}(U) + T_{3,t}(U) \]

\[ = U \]

\[ + t(A(U) + \frac{1}{2} H(U)) \]

\[ + t^2(B(U) + \frac{1}{2} H^{(1)}(U)(A(U) + \frac{1}{4} J(U))) \]

\[ + C_{t}(U) \]

\[ = U \]

\[ + t(M(U) + H(U)) \]

\[ + \frac{t^2}{2}(M + H^{(1)}(U))(M + H(U)) \]

\[ + C_{t}(U), \]

where

\[ \lim_{t \downarrow 0} t^{-2} ||C_t(U)||_s = 0 \quad \forall U \in H_{s+4}, \]

\[ \lim_{t \downarrow 0} t^{-3} ||C_t(U)||_s = 0 \quad \forall U \in H_{s+6}. \]
2.5 Operator Splitting

\[
\frac{d}{dt} \begin{bmatrix} u^{(R)} \\ v^{(R)} \end{bmatrix} = \begin{bmatrix} F(u^{(R)}, v^{(R)}) \\ G(u^{(R)}, v^{(R)}) \end{bmatrix}, \quad 0 < x < 1, \quad 0 < t < \Delta t/2, \tag{20a}
\]

\[
u^{(R)}(x, 0) = u(x, 0), \quad v^{(R)}(x, 0) = v(x, 0), \quad 0 < x < 1. \tag{20b}
\]

\[
\frac{\partial u^{(E)}}{\partial t} = \frac{\partial}{\partial x} \left( D \frac{\partial u^{(D)}}{\partial x} \right), \quad 0 < x < 1, \quad 0 < t < \Delta t, \tag{21a}
\]

\[
u^{(D)}(x, 0) = u^{(R)}(x, \Delta t/2), \quad 0 < x < 1. \tag{21b}
\]

\[
\frac{d}{dt} \begin{bmatrix} u^{(R)} \\ v^{(R)} \end{bmatrix} = \begin{bmatrix} F(u^{(R)}, v^{(R)}) \\ G(u^{(R)}, v^{(R)}) \end{bmatrix}, \quad 0 < x < 1, \quad \Delta t/2 < t < \Delta t, \tag{22a}
\]

\[
u^{(R)}(x, \Delta t/2) = u^{(D)}(x, \Delta t), \quad v^{(R)}(x, \Delta t/2) = v^{(D)}(x, \Delta t), \quad 0 < x < 1. \tag{22b}
\]

3 Discretization Techniques

3.1 Overview of the Method

3.1.1 Preliminary Considerations

We have several useful observations to consider while developing our numerical approach to the solution of the Fitzhugh-Nagumo problem. First, we note that at spatial locations where the stiff nonlinear reactions cause rapid change in the solution, it is useful to take small timesteps to preserve some overall desired accuracy in the results. Secondly, it is useful to use an unconditionally stable discretization of the diffusion in order to avoid severe restrictions on the size of the timestep; such discretizations determine the numerical solution implicitly. However, the timesteps needed for accuracy in the numerical discretization of the reaction may be very different from the timesteps needed for accurate and stable treatment of the diffusion.

Implicit treatment of diffusion necessarily couples all unknowns in space. In order to simplify the numerical solution of the implicit system, it is useful to separate the stiff but spatially uncoupled nonlinear reaction from the spatially-coupling diffusion. Reaction by itself leads to uncoupled nonlinear ordinary differential equations at the individual spatial locations in the grid; these ordinary differential equations can be integrated with individual choices for the timestep. Diffusion by itself leads to a symmetric positive-definite system of linear equations; these linear equations can be solved by fast iterative methods in multiple dimensions.

Excitable media equations such as the Fitzhugh-Nagumo model lead to propagating wave fronts. In order to avoid unphysical pinning of the numerical wave front, it is necessary to use sufficient mesh refinement in a neighborhood of the wave front. In order to produce a numerical wave that travels with the correct front speed, it may be necessary to use even finer mesh in the neighborhood of the front. This fine mesh is not necessary away from the front, and uniform grid calculations can waste a large amount of effort away from the front. Thus, it is desirable to adapt the mesh dynamically, in both space and time.

The use of adaptive mesh refinement places certain restrictions on the numerical components of the algorithm. For example, we do not know how to use alternating-direction-implicit (ADI)
methods with adaptive mesh refinement, since these methods are basically spatial operator splitting techniques and our spatial adaptation in multiple dimensions may not be efficient if it is defined solely with respect to the coordinate axes. As another example, we note that it is difficult to use high-order multistep algorithms (such as backward differentiation formulas) with adaptive mesh refinement because these require initial data from several previous timesteps; during regridding steps for adaptive mesh refinement it would be necessary to create this initial data artificially by high-order temporal and spatial interpolation from coarse grid data.

The regridding process in adaptive mesh refinement introduces another complication. In order to keep the cost of regridding acceptably low, we regrid at predetermined numbers of timesteps on coarse grids and place grid refinement in advance of the front; in order to keep the front within the refinement until the next regridding event we need to have reliable estimates of the speed of the front. In practical application to models of electrical activity in the heart this is not a problem, because the speed of the wave front is known.

A more serious difficulty with adaptive mesh refinement lies in designing spatial discretizations of the diffusion equation that lead to symmetric positive-definite linear systems on grids that are composites of cells in a hierarchy of refinement. We will describe our successful design in sections 3.3 and 5.1.

3.1.2 Numerical Approach

Our approach will be to use operator splitting to separate the reaction from the diffusion. In order to obtain overall second-order accuracy, we will use a half-step of reaction, followed by a full step of diffusion and another half-step of reaction. The reaction will be integrated by a second-order Stiff Diagonally-Implicit Runge-Kutta (SDIRK) scheme. These reactions are uncoupled, and the solution of their implicit equations is straightforward using analytical solutions for cubic equations. The diffusion will be discretized in space by a second-order conforming finite element method, and in time by the Crank-Nicolson method. The linear systems for the diffusion can be solved directly in one dimension for non-adaptive grids. With adaptive mesh refinement, we use conjugate gradient iterations preconditioned by multigrid V-cycles. These numerical techniques extend nicely to multiple dimensions.

3.2 Integration of Reactions

Operator splitting of the Fitzhugh-Nagumo problem leads to two sub-problems (20) and (22) for the reaction, each consisting of a system of ordinary differential equations at each point in space. For the Fitzhugh-Nagumo model, there are two equations in this system. However, for more realistic models there might be dozens.

3.2.1 Generic Form of the SDIRK Scheme

To integrate these ordinary differential equations, we will use a second-order SDIRK scheme, described in [19]. To simplify the discussion of the SDIRK scheme in this section, we will write the system of ordinary differential equations in the generic form

$$\frac{\partial y}{\partial t} = \phi(y), \quad y(0) = y_0.$$

Then the SDIRK scheme involves numerical approximations $y^n \approx y(n \Delta t)$, where the discrete solution $y^n$ is defined by
\[
\begin{align*}
k_1 &= \phi(y^n + k_1 \gamma \Delta t) \\
k_2 &= \phi(y^n + (1 - 2 \gamma)k_1 \Delta t + \gamma k_2 \Delta t) \\
y^{n+1} &= y^n + (k_1 + k_2) \frac{\Delta t}{2}.
\end{align*}
\] (23a)  
(23b)  
(23c)

Here $\gamma \equiv 1 - \sqrt{1/2}$. Notice that the equations (23a) and (23b) define $k_1$ and $k_2$ implicitly. We will discuss our approach to this complication later in section 3.2.5.

### 3.2.2 Truncation Error

An analysis of this scheme shows that the Taylor expansion for the computed solution gives
\[
y^{n+1} \approx y_n + fn \Delta t + f' f \frac{\Delta t^2}{2} + (f')^2 f \gamma (1 - \gamma) \Delta t^3 + f'' f^2 (1 - 2 \gamma + 2 \gamma^2) \frac{\Delta t^3}{4}.
\]

Since a Taylor expansion of the true solution gives
\[
y(t^n + \Delta t) \approx y(t^n) + f_n \Delta t + f' f \frac{\Delta t^2}{2} + [(f')^2 f + f'' f^2] \frac{\Delta t^3}{6},
\]
we would need to require that
\[
\gamma (1 - \gamma) = \frac{1}{6} \quad \text{and} \quad (1 - 2 \gamma + 2 \gamma^2) \frac{1}{4} = \frac{1}{6} \quad \implies \gamma = \frac{3 \pm \sqrt{3}}{6}.
\]
to obtain third-order accuracy. However, this more accuracy choice of $\gamma$ does not have the best stability properties.

### 3.2.3 L-Stability

With the choice $\gamma \equiv 1 - \sqrt{1/2}$, this SDIRK scheme is L-stable. Technically, this means that when we apply the SDIRK scheme to the differential equation $y' = \lambda y$, we get
\[
y^{n+1} = y_n R(\lambda \Delta t)
\]
where
\[
R(\lambda \Delta t) \equiv 1 + \frac{\lambda \Delta t}{1 - \gamma \lambda \Delta t} + \frac{1 - 2 \gamma}{2} (\frac{\lambda \Delta t}{1 - \gamma \lambda \Delta t})^2
\] (24)
\[
\to 1 - \frac{1}{\gamma} + \frac{1 - 2 \gamma}{2 \gamma^2} = (1 - \frac{1 + \sqrt{1/2}}{\gamma})(1 - \frac{1 - \sqrt{1/2}}{\gamma}) = 0 \quad \text{as} \quad \lambda \Delta t \to \infty.
\]
3.2.4 A-Stability

With the choice $\gamma = 1 - \sqrt{1/2}$, this SDIRK scheme is also A-stable. Technically this means that for all $\lambda \Delta t$ with negative real part, the amplification factor $R(\lambda \Delta t)$ defined in (24) satisfies $|R(\lambda \Delta t)| < 1$. To show that the SDIRK scheme is A-stable, we will use the following argument. If $z = \lambda \Delta t$, define the linear fractional transformation

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

It is easy to see that $w$ maps the left half-plane into the interior of the circle $|2\gamma w + 1| \leq 1$. If $w = (\zeta - 1)/(2\gamma)$ where $\zeta = \cos(\psi) + i\sin(\psi)$ then it is straightforward to see that

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

The extreme points of this function occur at $\psi = 0, \pi$ and $2\pi$, where $|R| = 1, 0$ and $1$, respectively. This shows that $|R(z)| \leq 1$ for all $z$ satisfying $|2\gamma w(z) + 1| \leq 1$, which in turn shows that $R$ maps the left half-plane into a subset of the unit circle.

3.2.5 Application of SDIRK to FitzHugh-Nagumo

In order to implement the SDIRK scheme for the FitzHugh-Nagumo model, we have adopted a special strategy. To illustrate the approach, we will consider the first implicit equation (23a) in the SDIRK scheme. We will deal with the $v$-component of the equation for $k_1$ first. Using the form of the function $G$ defined in (2b) we see that this equation takes the form

$$k_v = \alpha H(u^n + \gamma k_n \Delta t) - \beta (v^n + \gamma k_v \Delta t).$$

We solve this linear equation to get

$$k_v(k_v) \equiv \frac{H \alpha (u^n + \gamma k_v \Delta t) - \beta v^n}{1 + \beta \gamma \Delta t}. \tag{25}$$

This defines $k_v$ as a linear function of $k_v$. Then we consider the cubic equation

$$k_v = HF(u^n + \gamma k_v \Delta t, v^n + \gamma k_v(k_v) \Delta t),$$

where $F$ is defined in (2a).

It is not hard to see that if $\Delta t$ is small enough, then there is exactly one real root of this cubic equation for $k_v$. In order to test if there is only one real root, it is convenient to replace $k_v$ in the cubic equation

$$A_0 k_v^3 + A_1 k_v^2 + A_2 k_v + A_3 = 0$$

by the expression $k_v = x - \frac{A_1}{3A_0}$ and obtain the equation

$$x^3 + 3B_2 x + 2B_3 \equiv x^3 + \left(\frac{A_2}{A_0} - \frac{A_1^2}{3A_0^2}\right)x + \frac{A_3}{A_0} - \frac{A_1 A_2}{3A_0^2} + \frac{4A_1^3}{27A_0^3} = 0.$$  

It is well-known that this cubic has exactly one real root if and only if $0 < B_2^2 + B_3^2$. If there were more than one real root, it would be necessary to select that root which tends to the correct value as $\Delta t \to 0$. In order to avoid this difficulty, we repeatedly reduce the timestep by a factor of 2 until both cubics (for the $u$-components of $k_1$ and $k_2$) have exactly one real root. After we solve either cubic equation for its $u$-component $k_u$, we can evaluate the $v$-component using (25).
3.3 Integration of Diffusion

To integrate the diffusion equation in the operator-splitting subproblem (21) we use a conforming finite element method. An excellent reference for this technique is [58]. We will summarize the method and error analysis here.

3.3.1 Semi-Discrete Galerkin Methods

The finite element method is designed to work with the diffusion problem in the variational form

\[ \forall w \in H^1(\Omega), \quad 0 = \frac{d}{dt} \int_{\Omega} w(x) u(x,t) \, dx + \int_{\Omega} \nabla_x w \cdot D \nabla_x u \, dx \]

\[ \forall x \in \Omega, \quad u(x,0) = u_C(x) . \]

Here \( H^1(\Omega) \) represents the usual Sobolev space, consisting of functions whose derivatives of order up to one are square-integrable on \( \Omega \). This problem is equivalent to (21) provided that the initial data are sufficiently smooth.

A semi-discrete Galerkin approximation to the solution of (26) begins by choosing a finite dimensional subspace \( \mathcal{M} \subset H^1(\Omega) \). We want to find a function \( U(x,t) \in \mathcal{M} \) for all \( t > 0 \) so that

\[ \forall W \in \mathcal{M}, \quad 0 = \frac{d}{dt} \int_{\Omega} W(x) U(x,t) \, dx + \int_{\Omega} \nabla_x W \cdot D \nabla_x U \, dx \]

\[ \forall W \in \mathcal{M}, \quad \int_{\Omega} (U(x,0) - u_0(x))^2 \, dx \leq \int_{\Omega} (W(x) - u_0(x))^2 \, dx . \]

A fully discrete Galerkin approximation uses an appropriate stiffly stable method for solving initial value problems for ordinary differential equations.

3.3.2 Crank-Nicolson Finite Element Method

A finite element method for approximating the solution of (26) is a Galerkin method that chooses the finite-dimensional subspace \( \mathcal{M} \) to consist of piecewise polynomials. In our application, we will choose \( \mathcal{M} \) to consist of continuous piecewise linear functions on some given mesh

\[ 0 = x_0 < x_1 < \ldots < x_N = 1 . \]

We will denote the element widths by

\[ \forall 0 \leq k < N \quad \Delta x_{k+\frac{1}{2}} \equiv x_{k+1} - x_k . \]

We will also use the trapezoidal rule to integrate in time; this leads to the familiar Crank-Nicolson method for the diffusion equation. It is known [58] that the error in this method satisfies

\[ \| u(\cdot, t) - U(\cdot, t) \|_{H^0(\Omega)} \leq C \epsilon^{-\lambda_1} \| u(\cdot, 0) - U(\cdot, 0) \|_{H^2(\Omega)} + \Delta x^2 \int_0^t \| \frac{\partial u}{\partial t}(\cdot, s) \|_{H^2(\Omega)} \, ds \]

\[ + \Delta t^2 \int_0^t \| \frac{\partial^3 u}{\partial t^2} \|_{H^1(\Omega)} \, ds \]
\[ + \Delta t^2 \int_0^t \left[ \frac{\partial^2 \nabla_x \cdot D \nabla_x u}{\partial t^2}(\cdot, s) \right] |f(\Omega)| ds \].

Here \( \lambda_1 \) is the smallest eigenvalue of \( -\nabla_x \cdot D \nabla_x \) on \( \Omega \). The scheme is basically second-order in space and time for \( f = O(1) \); 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.

### 3.3.3 Numerical Implementation

It is possible to implement the finite element method so that it has the form of a finite difference method. The advantages of the finite difference form are both readability and computational efficiency. The advantages of the finite element foundation are provable convergence estimates in fairly general circumstances, and straightforward application to multigrid iteration.

Given the mesh \( \{ B \} \), we will use isoparametric transformations to define the basis functions for the continuous piecewise-linear functions in our finite element space \( \mathcal{M} \). We define the canonical basis function (sometimes called the "hat function")

\[
V_\ell(x) \equiv \max\{1 - |x||, 0\}.
\]

We also define the piecewise linear coordinate mappings

\[
\forall 0 \leq \ell \leq N, \quad \ell(x) \equiv \begin{cases} 
(x - x_\ell)/(x_{\ell+1} - x_\ell), & x \geq x_\ell \\
(x - x_\ell)/(x_\ell - x_{\ell-1}), & x \leq x_\ell
\end{cases}
\]

(We can ignore the unnecessary cases for \( \ell = 0 \) and \( \ell = N \).) Note that we can invert this relation to get a mapping from \( x \in (0, 1) \) to \( x \in (x_j, x_{j+1}) \):

\[
x(\xi) = x_j + \xi \Delta x_{j+1} \quad x \in (0, 1)
\]

Then our continuous piecewise linear basis functions are

\[ V_j(x) \equiv V(\ell(x)), \forall 0 < j < N. \]

The finite element equations lead to the linear system for \( 0 \leq j \leq N \),

\[
0 = \sum_{j=0}^N \sum_{\ell=0}^{j-1} \left\{ \int_{x_\ell}^{x_{\ell+1}} \frac{\partial V_j}{\partial x} (x_{\ell+1}) \frac{\partial V_k}{\partial x} (x_\ell) \right\} dx \left[ u_{\ell+1} - u_j \right] + \Delta t^2 \int_{x_\ell}^{x_{\ell+1}} \frac{\partial V_j}{\partial x} (x_{\ell+1}) \frac{\partial V_k}{\partial x} (x_\ell) \right\} dx \left[ \frac{u_{\ell+1}}{2} + u_j \right].
\]

The finite element equations say that for each basis function \( V_j \) the weak form of the equations must be satisfied. Note that the nonzero contributions to the mass or stiffness matrices from element \( (x_\ell, x_{\ell+1}) \) only involve \( V_\ell \) and \( V_{\ell+1} \). This suggests that we form the diffusive fluxes

\[
\begin{bmatrix}
\phi_{k+1/2, k} \\
\phi_{k+1/2, k+1}
\end{bmatrix} = 
\begin{bmatrix}
\int_{x_\ell}^{x_{\ell+1}} V_\ell(x_{\ell+1}) V_{\ell+1}(x) \frac{\partial V_k}{\partial x}(x_\ell) dx \\
\int_{x_\ell}^{x_{\ell+1}} V_\ell(x_{\ell+1}) V_{\ell+1}(x) \frac{\partial V_k}{\partial x}(x_\ell) dx
\end{bmatrix}
\]

\[ + \Delta t^2 \int_{x_\ell}^{x_{\ell+1}} \frac{\partial V_j}{\partial x} (x_{\ell+1}) \frac{\partial V_k}{\partial x} (x_\ell) \right\} dx \left[ \frac{u_{\ell+1}}{2} + u_j \right].
\]

The finite element equations in the interior of the domain then take the form

\[
\forall 0 < k < N, \quad \phi_{k+1/2, k} + \phi_{k+1/2, k} = 0.
\]
The Neumann boundary conditions are approximated by the additional two equations \( \dot{\phi}_{\frac{j}{2},L} = 0 \) and \( \dot{\phi}_{\frac{j}{2},N} = 0 \).

We can also write the finite element equations in matrix-vector form as the linear system

\[
0 = M[u^{n+1} - u^n] + K[u^{n+1} + u^n] \frac{\Delta t^{n+\frac{1}{2}}}{2} .
\]

It is easy to evaluate the mass matrix \( M \) and the stiffness matrix \( K \) analytically. In fact, it suffices to compute the element-wise contribution to the mass matrix:

\[
\int_{x_j}^{x_{j+1}} \begin{bmatrix} V_k & V_{k+1} \end{bmatrix} \begin{bmatrix} V_k & V_{k+1} \end{bmatrix} dx = \Delta x_{k+\frac{1}{2}} \int_{0}^{1} \begin{bmatrix} V(\xi) & 1 - V(\xi) \end{bmatrix} \begin{bmatrix} V(\xi) & 1 \end{bmatrix} d\xi = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} \end{bmatrix},
\]

and the element-wise contribution to the stiffness matrix

\[
\int_{x_j}^{x_{j+1}} \begin{bmatrix} \frac{\partial V_k}{\partial x} & \frac{\partial V_{k+1}}{\partial x} \end{bmatrix} D(x) \begin{bmatrix} \frac{\partial V_k}{\partial x} & \frac{\partial V_{k+1}}{\partial x} \end{bmatrix} dx = \frac{1}{\Delta x_{k+\frac{1}{2}}} \int_{0}^{1} \begin{bmatrix} -1 & 1 \\ 1 & -1 \end{bmatrix} D(\xi) \begin{bmatrix} -1 & 1 \\ 1 & -1 \end{bmatrix} d\xi
\]

where \( D_{k+\frac{1}{2}} \) is the average of the diffusion coefficient:

\[
D_{k+\frac{1}{2}} = \frac{\int_{x_k}^{x_{k+1}} D(x') dx'}{\Delta x_{k+\frac{1}{2}}}.
\]

These computations can then be used to evaluate the diffusive fluxes \( \dot{\phi}_{k+\frac{1}{2},j/k+1} \).

We can also write the finite element scheme in the form of a finite difference scheme:

\[
\forall 0 < j < N \quad 0 = \frac{\Delta x_{j+\frac{1}{2}}}{6} \left\{ (v_{j+1}^n - v_j^n) + 2(v_{j+1}^{n+1} - v_j^{n+1}) + \frac{\Delta x_{j+1}}{6} \{2(v_{j+1}^{n+1} - v_j^n)(v_{j+1}^{n+1} - v_{j+1}^n)\} \right\}
\]

\[
+ \frac{D_{j+\frac{1}{2}} \Delta t^{n+\frac{1}{2}}}{2 \Delta x_{j+\frac{1}{2}}} \left\{ (v_{j+1}^{n+1} - v_{j+1}^n) + (v_j^n - v_{j+1}^{n+1}) \right\}
\]

At the left boundary we have the equation

\[
0 = \frac{\Delta x_{\frac{1}{2}}}{6} \left\{ 2(v_0^{n+1} - v_0^n)(v_1^{n+1} - v_1^n) \right\} - \frac{D_{\frac{1}{2}} \Delta t^{n+\frac{1}{2}}}{2 \Delta x_{\frac{1}{2}}} \left\{ (v_1^{n+1} - v_0^{n+1}) + (v_1^n - v_0^n) \right\}
\]

and at the right boundary we have the equation

\[
0 = \frac{\Delta x_{N-\frac{1}{2}}}{6} \left\{ (v_{N-1}^{n+1} - v_{N-1}^n) + 2(v_{N}^{n+1} - v_{N}^n) \right\} + \frac{D_{N-\frac{1}{2}} \Delta t^{n+\frac{1}{2}}}{2 \Delta x_{N-\frac{1}{2}}} \left\{ (v_{N}^{n+1} - v_{N-1}^{n+1}) + (v_{N}^n - v_{N-1}^n) \right\}.
\]

Note that the weighted averaging of the time derivatives is different from a standard finite difference scheme; this weighting will have advantages when we describe the multigrid method below.
3.3.4 Linear Algebra on a Non-Adaptive Grid

The linear system \(30\) can be written in the form

\[
(M + K \frac{\Delta t}{2})u^{n+1} = (M - K \frac{\Delta t}{2})u^n;
\]

this system is tri-diagonal. On a uniform grid in one dimension, this linear system is can be solved directly and rapidly by matrix factorization. On an adaptive grid in one dimension, the data structures may make it awkward to use a direct solver, especially if these data are on different processors of a distributed memory machine.

In multiple dimensions, the linear system is banded. Typically, direct solver methods are less efficient than iterative methods for such linear systems arising from the diffusion equation.

3.3.5 Timestep Selection

Monotonicity Preservation

Since the Crank-Nicolson scheme employs the \(A\)-stable trapezoidal rule for time integration, it is unconditionally stable, meaning that perturbations in the numerical solution decay with time. As a result, this notion of stability places no restriction on the timestep. In the absence of any other timestep restrictions, it would be natural to choose the timestep to balance the spatial and temporal truncation errors in the Crank-Nicolson scheme. Since this scheme is second-order in both space and time, this would imply that \(\Delta t = O(\Delta x)\).

However, the trapezoidal rule is not \(L\)-stable. For example, with very large \(\Delta t\) the linear system (31) tends to the equation \(u^{n+1} = -u^n\). However, there is a more useful way to understand the problem. If we decompose \(u^n\) in terms of the eigenvectors of \((M + K \frac{\Delta t}{2})^{-1}(M - K \frac{\Delta t}{2})\), then we will find that for large \(\Delta t\) the eigenvalues will lie between -1 and 1. This implies that the expansion coefficients corresponding to the large eigenvalues will alternate in sign between timesteps. This leads to numerical oscillations that decay in time. Such numerical oscillation can be avoided by choosing \(\Delta t\) sufficiently small so that \(M - K \frac{\Delta t}{2}\) is positive definite.

The Gerschgorin circle theorem [2, 32] implies that a symmetric matrix is positive-definite whenever each diagonal entry is greater than the corresponding sum of the absolute values of the off-diagonal entries in the same row or column. The matrix-vector form of the finite element equations (31) makes it easy for us to see that the matrix \(M - K \frac{\Delta t}{2}\) multiplying \(u^n\) is positive-definite whenever

\[
\frac{\Delta x_{k-\frac{1}{2}}}{3} + \frac{\Delta x_{k+\frac{1}{2}}}{3} - \frac{D_{k-\frac{1}{2}} \Delta t^{n+\frac{1}{2}}}{\Delta x_{k-\frac{1}{2}}/2} - \frac{D_{k+\frac{1}{2}} \Delta t^{n+\frac{1}{2}}}{\Delta x_{k+\frac{1}{2}}/2} > \frac{\Delta x_{k-\frac{1}{2}}}{6} + \frac{\Delta x_{k+\frac{1}{2}}}{6} + \frac{D_{k-\frac{1}{2}} \Delta t^{n+\frac{1}{2}}}{\Delta x_{k-\frac{1}{2}}/2} + \frac{D_{k+\frac{1}{2}} \Delta t^{n+\frac{1}{2}}}{\Delta x_{k+\frac{1}{2}}/2},
\]

This implies that

\[
\forall \ 0 < k < N, \ \Delta t < \frac{1}{6 \frac{D_{k-\frac{1}{2}}}{\Delta x_{k-\frac{1}{2}}} + \frac{D_{k+\frac{1}{2}}}{\Delta x_{k+\frac{1}{2}}}} \frac{\Delta x_{k-\frac{1}{2}} + \Delta x_{k+\frac{1}{2}}}{\Delta t}. \tag{32}
\]

At the endpoints, we have the additional restrictions

\[
\Delta t < \frac{\Delta x_{1}}{6D_{1/2}} \quad \text{and} \quad \Delta t < \frac{\Delta x_{N-1}}{6D_{N-1/2}}.
\]
The small timestep in (32) for monotonicity preservation is not necessary for \( t > 0 \), or for smooth initial data. Since eigenmodes corresponding to large eigenvalues of \((M + K\Delta t)^{-1}(M - K\Delta t)\) decay very rapidly when \( \Delta t \) is chosen to make this matrix positive definite, it does not matter if these rapidly decayed amplitudes oscillate at later time.

**Front Propagation in AMR**

However, adaptive mesh refinement will place an additional restriction on the timestep. In section 4.2 we will see that it will be useful to choose the timestep so that the reaction front does not move more than one grid cell per timestep. Based on our results in section 2.2.4, this will require that inside the reaction front

\[
\Delta t^{n+\frac{1}{2}} < \frac{\Delta x^{n+\frac{1}{2}}}{\sqrt{2}(\bar{u}_+ + \bar{u}_-)/2 - \bar{u}_+}}.
\]

We will approximate this timestep restriction by

\[
\Delta t^{n+\frac{1}{2}} < \frac{\Delta x^{n+\frac{1}{2}}}{\sqrt{2}(\frac{1}{2} - \theta)};
\]

(33)
corresponding to the assumption that the gating variable is nearly zero inside the reaction front.

**Comparison to Explicit Timestepping**

If we used explicit timestepping with a forward Euler time integration, we would be restricted by stability considerations to choose

\[
\Delta t < \frac{\Delta x^2}{2D} = \frac{H\Delta x^2}{2}.
\]

Our timestep selection leads to a larger timestep than explicit time integration whenever

\[
\Delta x < \frac{\sqrt{2}}{H(\frac{1}{2} - \theta)}.
\]

Recall from (16) that it is useful to choose

\[
\Delta x < \frac{2}{H\theta}
\]

to avoid pinning of the reaction front, and \( \Delta x \) even smaller to produce an accurate front speed. Finally, recall that \( \Delta x \) must be chosen even smaller than that to produce an accurate front speed. Thus our implicit timestepping strategy always chooses a larger timestep than a stable explicit scheme, when both avoid pinning.

**3.3.6 Relative Efficiency of Explicit and Implicit Time Integration**

It is common to use explicit time integration in numerical methods for electrical wave propagation in the heart. The reason for this choice is that it is difficult to design linear solvers that make implicit time integration efficient.

Suppose that we have a choice of two schemes for solving a linear time-dependent partial differential equation: an implicit scheme choosing \( \Delta t = O(\Delta x^\delta) \) in order to balance spatial and temporal truncation errors, and an explicit scheme choosing \( \Delta t = O(\Delta x^2) \) for stability. Also suppose that the number of unknowns in the linear system system for the implicit scheme is \( M = O((1/\Delta x)^\delta) \), where \( \delta \) is the number of spatial dimensions in the problem. Finally, suppose that the implicit method
solves the linear system by means of an iterative process that costs $O(M)$ operations per iteration and on the order of $M^p$ iterations to reach the same order of accuracy as the explicit scheme. Then in order for the implicit scheme to be more efficient than the explicit scheme as $\Delta x \to 0$, we will show that we must have $p < (2 - r)/\delta$.

Let us justify this result. Because of its stability restriction, the explicit scheme will take on the order of a factor of $(1/\Delta x)^{2-r}$ more timesteps to catch up with a single implicit timestep. The total work for the explicit scheme to take enough timesteps to catch up with one timestep of an implicit scheme is on the order of $O((1/\Delta x)^{5+2-r})$. On the other hand, the total work in one timestep of the implicit method is the work per iteration times the number of iterations, or $O(MM^p) = O((1/\Delta x)^{5(1+p)})$. In order for the implicit method to require a lower order of work than the explicit method, we require its work to have a smaller exponent: $\delta(1 + p) < \delta + 2 - r$. This is equivalent to the claim.

This result says that the number of iterations in the implicit solver should be at worst $M^p = (1/\Delta x)^{5p} \leq O((1/\Delta x)^{2-r})$. In particular, note that if the spatial order of the implicit scheme is twice the temporal order ($r = 2$), as it is for centered differences in space and backward Euler in time, then the iterative method must use fewer than $O(1)$ iterations per timestep in order to be asymptotically more efficient than explicit time integration. This argues strongly against first-order implicit time integration.

With our choice of the Crank-Nicolson scheme and the timestep selection strategy (33) we have $r = 1$. Thus for the method described in this paper we want to design a linear solver to use fewer than $O(1/\Delta x)$ iterations, with each iteration involving work proportional to the total number of unknowns.

In one dimension on a non-adaptive grid, we can use a direct solver. Since the linear system is tridiagonal in this case, the direct solver uses $O(M)$ work, and finds the solution in one iteration. In this case, it is easy for implicit Crank-Nicolson time integration to take less work than an explicit scheme in order to reach some given time.

In multiple dimensions, conjugate gradient iterations typically take $O(1/\Delta x)$ iterations to converge. In this case, it becomes difficult to predict whether implicit time integration will take less work than explicit time integration. Thus, it is desirable to find an iterative method that is significantly faster than conjugate gradients. Our approach will be to use a multigrid iteration to solve the linear equations.

### 3.4 Multigrid Iteration

It is known [58] that each iteration of the multigrid method for the heat equation involves work proportional to the total number of unknowns, and achieves a reduction in the error that is (roughly, up to factors of the logarithm of the number of unknowns) independent of the number of unknowns. As a result, the total work required to solve a linear system by multigrid iteration is (roughly) proportional to the total number of unknowns. However, the implementation details for multigrid methods applied to parabolic equations are usually left to the reader. We will provide most of those details below.

#### 3.4.1 Overview of the Multigrid Algorithm

Suppose that we want to solve

$$A_f \mathbf{x}_f = b_f$$
on some fine grid discretization of a partial differential equation. Here equation (31) implies that 
\( A_f = M + K \lambda t/2 \) is symmetric and positive definite. If possible, we would like to approximate the 
solution of this equation by solving a related equation on a coarser grid

\[ A_c \tilde{x}_c - b_c. \]

Suppose that for each fine grid we are given a restriction \( R \) that maps vectors in the domain of \( A_f \) to 
vectors in the domain of \( A_c \), and a prolongation \( P \) that maps vectors in the domain of \( A_c \) to vectors 
in the domain of \( A_f \). We assume that \( R = P^T \beta \) for some scalar \( \beta \) that depends on the grid and the 
number of dimensions. We assume that the coarse matrix is determined from the fine matrix by

\[ A_c = RA_fP. \]

Also suppose that on each grid we are given a smoother \( S_f \approx A_f^{-1} \). Then a single iteration of the 
multigrid algorithm takes the form

if there exists a coarser grid
  if there is no finer level \( r_f = r_f^{(0)} = A_f x_f - b_f \)
    \( d_f = -S_f r_f \)
    \( r_f = A_f d_f + r_f^{(0)} \)
    \( d_c = 0 \)
    \( r_c^{(C)} = R r_f \)
  call coarser multigrid with initial residual \( r_c \), obtaining correction \( d_c \)
    \( d_f = P d_c \)
    \( r_f = A_f d_f + r_f^{(C)} \)
  \( d_f = S_f^T r_f \)
  if there is no finer level \( x_f^+ = d_f \)
else
    solve the coarsest linear system \( A_c d_c = r_c^{(C)} \)

We will discuss choices of the restriction, prolongation and smoother later in this section.

### 3.4.2 Multigrid Errors

By examining the multigrid algorithm, it is straightforward to see that at the completion of each 
level of the multigrid iteration (34), the error at iteration \( n \) satisfies

\[ A_f^{\frac{1}{2}}(x_f^{n+1} - \tilde{x}_f) = (I - A_f^{\frac{1}{2}} S_f^{T} A_f^{\frac{1}{2}})(I - A_f^{\frac{1}{2}} PV R A_f^{\frac{1}{2}})A_f^{\frac{1}{2}}(x_f^n - \tilde{x}_f). \]

Here \( V \) represents the effect of the recursive call of the algorithm on coarser grids.

The form of this error expression indicates several important aspects of the algorithm. The outer 
factor \( I - A_f^{\frac{1}{2}} S_f^T A_f^{\frac{1}{2}} \) and its transpose approximately project the error into the subspace where \( S_f \) 
does not approximate \( A_f^{-1} \) well. If the smoother is designed properly, this subspace is approximately 
annihilated by the inner term \( I - A_f^{\frac{1}{2}} PV R A_f^{\frac{1}{2}} \). Since the restriction \( R \) maps from the fine solution 
space to the (presumably much smaller) coarse solution space, its nullspace is nonzero. Thus, we 
would like the nullspace of \( RA_f^{\frac{1}{2}} \) to be roughly complementary to the subspace where \( I - A_f^{\frac{1}{2}} S_f A_f^{\frac{1}{2}} \) is 
small. The details of this interplay between the smoother and the restriction are covered in standard 
fine element treatments of multigrid methods; see [11, 14, 58] for details.
Sometimes this relationship is explained heuristically as follows: the fine grid smoother reduces the high-frequency components of the error, which lie in the null-space of the restriction. The remaining low-frequency components of the error on the fine grid look more like high-frequency components on the coarse grid. As a result, the recursive coarse grid computations are able to reduce the remaining components of the error. The recursion serves the purpose of reducing the errors on all scales while keeping the total work proportional to the total number of unknowns.

3.4.3 Restriction and Prolongation

The restriction and prolongation operators are adjoints of each other, in the following sense:

\[ \forall u, \forall v_c \quad v_c^T (Ru) = (Pv_c)^T u_f. \]

Note that this relation implies that \( R = P^T \) in the sense of matrices. Here we have omitted the factor of the refinement ratio found in [14], because we are trying to maintain the discrete equations in a form that corresponds directly to the finite element quadratures.

With conforming finite element methods using nested function spaces for the finite element approximations on neighboring levels of refinement, prolongation is naturally defined by injection of the coarse function into the fine space. Once the finite element approximations are written as linear combinations of the basis functions on the different levels of refinement, we obtain the following formula for the unknowns on the fine level \( k \) of refinement, in terms of the unknowns on the coarse level \( k - 1 \):

\[
    u_i^{(k)} = \begin{cases} 
        u_i^{(k-1)}, & i = 2I \\
        \frac{1}{2}(u_i^{(k-1)} + u_{i+1}^{(k-1)}), & i = 2I + 1
    \end{cases}
\]

This formula amounts to linear interpolation of the coarse results on the fine grid.

The adjoint of this prolongation is the restriction

\[
    u_f^{(k-1)} = \frac{1}{2}u_{2I}^{(k)} + u_{2I}^{(k)} + \frac{1}{2}u_{2I+1}^{(k)}.
\]

It is straightforward to find the coarse grid matrix \( A^{(k-1)} = RA^{(k)}P \). Note that

\[
    (A^{(k)}u^{(k)})_i = \left( \frac{\Delta x_i}{6} - \frac{D_{i-1/2} \Delta t^{n+1/2}}{2 \Delta x_i} \right) u_{i-1}^{(k)} + \left( \frac{\Delta x_i}{3} + \frac{\Delta x_{i+1}}{3} - \frac{D_{i-1/2} \Delta t^{n+1/2}}{2 \Delta x_i} - \frac{D_{i+1/2} \Delta t^{n+1/2}}{2 \Delta x_i} \right) u_i^{(k)} + \left( \frac{\Delta x_{i+1}}{6} - \frac{D_{i+1/2} \Delta t^{n+1/2}}{2 \Delta x_{i+1}} \right) u_{i+1}^{(k)}.
\]

By choosing \( v^{(k-1)} \) to be an axis vector in the equation

\[
    (v^{(k-1)}, A^{(k-1)}u^{(k-1)})_{k-1} - (Pv^{(k-1)}, A^{(k)} Pu^{(k-1)})_k
\]
we see that

\[
(A^{(k-1)}u^{(k-1)})_I = \left(\frac{\Delta x_{2I-\frac{1}{2}}}{6} + \frac{\Delta x_{2I+\frac{1}{2}}}{6} - \frac{D_{2I-\frac{1}{2}}\Delta t^{n+\frac{1}{2}}}{8\Delta x_{2I-\frac{1}{2}}} - \frac{D_{2I+\frac{1}{2}}\Delta t^{n+\frac{1}{2}}}{8\Delta x_{2I+\frac{1}{2}}}\right)u^{(k-1)}_I \\
+ \left(\frac{\Delta x_{2I-\frac{1}{2}}}{12} + \frac{7\Delta x_{2I-\frac{1}{2}}}{12} + \frac{7\Delta x_{2I+\frac{1}{2}}}{12} + \frac{\Delta x_{2I+\frac{1}{2}}}{12}\right)u^{(k-1)}_I \\
+ \left(\frac{D_{2I-\frac{1}{2}}\Delta t^{n+\frac{1}{2}}}{8\Delta x_{2I-\frac{1}{2}}} + \frac{D_{2I+\frac{1}{2}}\Delta t^{n+\frac{1}{2}}}{8\Delta x_{2I+\frac{1}{2}}} + \frac{D_{2I-\frac{1}{2}}\Delta t^{n+\frac{1}{2}}}{8\Delta x_{2I+\frac{1}{2}}} + \frac{D_{2I+\frac{1}{2}}\Delta t^{n+\frac{1}{2}}}{8\Delta x_{2I+\frac{1}{2}}}\right)u^{(k-1)}_I \\
+ \left(\frac{\Delta x_{2I-\frac{1}{2}}}{6} + \frac{\Delta x_{2I-\frac{1}{2}}}{6} - \frac{D_{2I-\frac{1}{2}}\Delta t^{n+\frac{1}{2}}}{8\Delta x_{2I-\frac{1}{2}}} - \frac{D_{2I+\frac{1}{2}}\Delta t^{n+\frac{1}{2}}}{8\Delta x_{2I+\frac{1}{2}}}\right)u^{(k-1)}_I
\]

On a uniform grid with constant diffusion, since the coarse cell width is \(\Delta x^{(k-1)} = 2\Delta x^{(k)}\), the coarsened operator for multigrid is identical with the original finite element equations on the coarse grid. If the mesh is not uniform or the diffusion is not constant, then it is necessary to compute \(A^{(k-1)}\) as above.

3.4.4 Smoother

We have chosen to use a Jacobi iteration for the smoother in multiple dimensions. For the residual \(r = Ax - b\), this iteration takes the form

\[
\text{for } 0 \leq j \leq N, \ x_j = x_j - r_j/A_{jj}.
\]

For the multigrid algorithm in one dimension, we use a direct solver as the smoother. (This choice will be possible, even with adaptive mesh refinement data structures, because the direct solver will be applied only to individual grid patches on a single level of refinement.)

As an initial guess for the linear system, we use the solution at the previous time. Typically, this initial guess leads to fairly small initial residuals. The multigrid iteration then reduces the residuals by a factor of 4-5 each iteration, so that 10 digits of accuracy are usually reached in at most 10 iterations, no matter how fine the grid may be.

As we have previously mentioned, it is very efficient to use a tridiagonal matrix solver for the linear system in one dimension on a uniform grid, more efficient than multigrid. However, the multigrid algorithm is very useful in solving a system of linear equations described on a hierarchical grid that arises from adaptive mesh refinement. Further, our implementation of the multigrid algorithm will allow us to implement the algorithm efficiently for distributed computing.

4 Adaptive Mesh Refinement

There are various forms of adaptive mesh refinement, ranging from static refinement [4, 55] to dynamic dynamic refinement in space [12, 13, 17, 22, 23, 24, 25, 26, 27, 28] and to dynamic refinement in both space and time [8, 10, 15, 35, 59].

In this paper, we will describe an adaptive mesh refinement 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 coarser synchronization. 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 computational organizational unit. Communication between patches is regular and small compared to the work within the patch, so the algorithm extends nicely to distributed computation.

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.

4.1 Basic Assumptions in Adaptive Mesh Refinement

In this section, we will summarize the basic principles of our adaptive mesh refinement algorithm. We assume that we are given some initial coarse mesh. We will call this mesh the coarsest level of refinement, and denote it by \( L_0 \). Imagine finer levels \( L_\ell \) defined recursively from the coarsest level \( L_0 \) according to the following rules.

**Finite Termination** In order to guarantee that the algorithm terminates, we require that there be a maximum number of levels, and a maximum number of timesteps on each level.

**Logical Rectangularity** We assume that each level \( L_\ell \) 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.

**Grid Alignment** 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 \( L_\ell \) coincides with the boundary of a logically rectangular array of coarse grid cells on level \( L_{\ell-1} \).

**Fixed Refinement Ratio** We assume that we are given an integer refinement ratio \( r \). Whenever a coarse cell on level \( L_\ell \) is refined, it is subdivided into \( r \) cells in each logical coordinate direction on level \( L_{\ell+1} \). Normally, the refinement ratio is a power of 2, rarely other than 2 or 4. Note that the assumptions of a fixed refinement ratio and of grid alignment imply that on any level \( L_\ell \) with \( \ell > 0 \), in any patch the number of cells in any coordinate direction is an integer multiply of the refinement ratio.

**Matched Temporal and Spatial Order** We assume that the integration scheme applied to the grid patches as well as to the communication between patches involves the same order of truncation error in both space and time. Thus refinement in space by a factor of \( r \) and refinement in time by a factor of \( r \) produce local truncation error that is approximately reduced by a factor of \( r^k \), where \( k \) is the spatial/temporal order. We assume that the scheme is stable when space and time are refined by the same ratio. (In this paper, we have designed the scheme to be second-order accurate in both space and time.) Note that matched temporal and spatial order is assumed merely as a matter of convenience; it simplifies the discussions of the error estimator and timestep selection.

**Necessity** To the greatest extent possible (consistent with the assumption of logical rectangularity), we want to locate the new fine mesh only where it is needed.
**Proper Nesting** We assume that the union of fine patches on level $L_{t+1}$ is contained in the interior of the union of coarse patches on level $L_t$. However, an individual fine patch is not required to lie inside any single coarse patch. Note that this assumption implies that the coarsest level $L_C$ 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 domain. This assumption can restrict the range of application of this form of adaptive mesh refinement. For modifications of the adaptive mesh refinement technique that extend to more general problem geometry, see [9, 7].

**Proper Nesting Buffer** We require that the boundary of the union of the fine patches be a fixed number of coarse cells from the boundary of the union of the coarser patches. This number of cells is called the proper nesting buffer. If chosen properly, it can be reasonably small and still guarantee that the interpolation needed to provide initial data on new fine cells does not need to recurse over coarser levels.

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

**Fine Preference** We assume that the numerical scheme for our differential equation produces better results on the fine grid than on the coarse, in all regions of the problem where both fine and coarse grid cells overlap.

**Conservation** We assume that the numerical scheme for our conservation law is conservative, whenever the underlying differential equations form a conservation law. Where fine and coarse grids overlap, we replace the coarse grid results with conservative coarsenings of the fine grid results.

**Infrequency of Regridding** We move the mesh after a fixed number of coarser timesteps, rather than after every coarse timestep. We assume that we are given a regrid interval, which is a predetermined number of timesteps between regridding events on a coarse level. At the end of each regrid interval, a coarse level $L_t$ selects coarse cells that need refinement at the new time, organizes these cells into some number of logically rectangular arrays of cells (usually by the addition of other coarse cells), and refines them to form the new cells on the finer level $L_{t+1}$. On any level $L_{t+1}$ that is not the finest level, the number of timesteps used for for synchronization with the coarser level $L_t$ must be an integer multiple of the regrid interval. This is required so that regridding can be applied recursively.

### 4.2 Timestep Selection

Now that we have stated our basic assumptions in the adaptive mesh refinement algorithm, we will describe the basic steps of the algorithm. The first task in integrating the data on the patches belonging to an arbitrary level $L_t$ is to select the current timestep $\Delta t_{c,t}$. If a coarser level exists, then the cell widths on the coarser level are a factor of the refinement ratio $r$ times the cell widths on this level. Thus, if a coarser level exists, the assumption of matched temporal and spatial order implies that the maximum number of steps on this level should be roughly $r$, so we should have that $r \Delta t_{c,t} \approx \Delta t_{c,t-1}$.

There are, of course, several modifications to this general rule. If the current level can be refined, then we have to take an appropriate number of timesteps so that we can perform Richardson error
estimation for regridding. There are two cases to consider: the coarsest level possible, and the intermediate levels rather than the finest possible.

If a coarser level does not exist, then we are working on the coarsest level, and we have no synchronization to perform. Assuming that this coarsest level \( C_0 \) can be refined, the number of timesteps we take to reach the desired time must be an integer multiple of the regrid interval. If necessary, we reduce \( \Delta t_{c,0} \) to reach the desired time at a regridding interval.

If a coarser level exists and we have already advanced a total time increment of \( \sum_{f=1}^{C_f} \Delta t_{f,t} \) on this level, then the time remaining until synchronization is \( \Delta t_{c,t-1} - \sum_{f=1}^{C_f} \Delta t_{f,t} \). Thus the number of timesteps remaining until synchronization with the coarser level is at least \( (\Delta t_{c,t-1} - \Delta t_{f,t})/\Delta t \). The maximum number of steps on this level is at least the number \( C_f \) of steps already taken plus this number of steps remaining. If the current level can be refined, then the maximum number of steps must be an integer multiple of the regridding interval. After increasing the number of timesteps remaining to enforce this condition, the updated value of the current timestep is the time remaining until synchronization divided by the number of steps remaining.

If the current level cannot be refined, then we can take as large a timestep as accuracy and stability allow. Since in this paper we are using an unconditionally stable scheme for the diffusion equation and a stiffly stable scheme with local time refinement for the reaction, it should be possible to take a single operator-splitting timestep on the finest grid.

If we have discontinuous initial data, it may be necessary to take small timesteps at early time to prevent numerical oscillations arising from the Crank-Nicolson scheme. This issue was discussed in section 3.3.5. These timesteps will be especially small on the finer levels.

It is generally a good idea to avoid rapid increases in the size of the timestep. This suggests that we require that the new timestep satisfy

\[
\Delta t_{f,t} \leq \min \{ \Delta t_{f-1,t}, \Delta x_t/\alpha \}
\]

where \( \Delta t_{f-1,t} \) is the previous timestep, \( \epsilon \) is the speed of the reaction front, and \( \alpha \) is some predetermined growth factor. Typically, \( \alpha = 1.1 \) is a good choice.

### 4.3 Recursive Timestepping

The algorithm to advance the data on the patches takes the form of the following algorithm:

```c
void Level::advance(double dt_max) {
    double dt_now=0, dt_total=0;
    findStableStepSize(dt_max, dt_total, dt_now);
    while (dt_total<dt_max) {
        reactAndSolveSystem(dt_now);
        if (finerLevelExists()) finerLevel()->advance(dt_now);
        if (timeToRegrid() && !isLastStep() && timeToRegridCcoarser())
            regridFinerLevels();
        dt_total += dt_now;
        findStableStepSize(dt_max, dt_total, dt_now);
    }
}
```

Let us repeat this algorithm in words. First, we determine the initial timestep size. 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. Afterward, we use recursion
to repeat the process on the finer levels. At appropriate times, we regrid the finer levels. Finally, we determine the next timestep size.

Two of these tasks, solving the linear system and regridding, require more elaboration.

4.4 Regridding

Because we are interested in solving time-dependent problems, we allow the mesh refinement to move in time. Here the adaptive mesh refinement principles of necessity, infrequency and proper nesting are important.

With respect to the necessity principle, we shall use an error estimation procedure (described in section 4.4.2 below) to determine where the unacceptably large errors occur on this level. By using an error estimator, rather than a gradient detector, we are able to place mesh refinement where discontinuities in the motion 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.

4.4.1 Proper Nesting

Recall that the proper nesting principle requires that the union of the fine patches be contained in the interior of the union of the coarser patches. This implies that the boundary of the union of the fine patches nowhere coincides with the boundary of the union of the coarser patches; in other words, if two neighboring grid cells are each on the finest possible level for their location in space, then these cells can differ by at most one refinement level. As a result, all interfaces between coarse and fine grids are related by a fixed ratio.

Think of the proper nesting list and the complement list as a list of boxes (i.e., patches without data). The complement list is initialized by finding a list of boxes that exactly cover the complement of the union of the patches with respect to the physical domain. Each of the boxes in this list is buffered to make them larger by a fixed number of cells, namely the proper nesting buffer. The proper nesting list is the list of boxes that forms the complement of the union of these buffered boxes.

Since we are moving the mesh after taking the regrid interval number of coarser timesteps, it is necessary to provide a buffer region around the cells currently requiring refinement. The purpose of the buffering is to prevent the waves of interest from moving off the refined mesh before the next regridding step. Here, we take advantage of the timestepping strategy in sections 3.3.5 and 4.2. In particular, the timestep has been chosen so that a wave can travel across at most one cell in a timestep.

Another purpose of the proper nesting list is to guide the selection of the patches on finer levels. The patches on the current level $\mathcal{L}_k$ have been selected so that the interesting waves will lie inside the union of the patches until the patches have been advanced in time to synchronization with the next coarser level $\mathcal{L}_{k-1}$. We want to construct a list of patches on each of the finer levels $\mathcal{L}_{k+1}, \ldots$ so that each fine patch lies inside the refinement of the proper nesting list belonging to its coarser level. When the union of patches on the current level $\mathcal{L}_k$ is not convex, the proper nesting list prevents patches on the next finer level $\mathcal{L}_{k+1}$ from straddling an interior corner of the union. This is not a problem in one dimension, but could be a problem in multiple dimensions.
4.4.2 Tagging Cells for Refinement

The next regrefield step is to update the list of patches on the finer level $L_{l+1}$. First, cells in the patches belonging to the current level $L_{l}$ 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 [18]. Suppose that at each timestep we commit an error of magnitude $\epsilon_t$ (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 (30) 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 $\epsilon_i^{(n)}$ in cell $i$ at step $n$ satisfies

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

An argument by induction shows that

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

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

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

where $k$ is the expected global order of the scheme. (In accordance with the assumption of matched temporal and spatial orders, we are assuming that spatial and temporal error orders are equal.) Then the error in taking one coarse step of size $r \Delta t$ is

$$\epsilon_{i,c}^{(n)} \approx Cr^{k+1}\Delta t^{k+1}.$$  

On the other hand, if we take $r$ fine timesteps of size $\Delta t$, the error is

$$\epsilon_{i,f}^{(n)} \approx Cr\Delta t^{k+1}.$$  

This allows us to estimate the local error of a fine timestep by

$$\epsilon_t \approx \frac{\epsilon_{i,c}^{(n)} - \epsilon_{i,f}^{(n)}}{r^{k+1} - r} = \frac{w_{i,c}^{(n)} - w_{i,f}^{(n)}}{r^{k+1} - r},$$

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 $N$. Here, $N \approx \frac{L}{r \Delta t}$, where $L$ is some length scale associated with the problem, $c$ is the speed of the reaction front (see section 2.2.4), and $\Delta t$ is the current timestep. Thus, cell $i$ is tagged for refinement if the relative error satisfies

$$\frac{|w_{i,c}^{(n)} - w_{i,f}^{(n)}|}{\max_j |w_{j,f}^{(n)}|} \frac{L}{(r^{k+1} - r)s\Delta t} > \text{tolerance}.$$
In summary the steps in the error estimation procedure are the following. Suppose that the regrid interval has the value \( n \). (In the algorithm described in this paper, we take \( n = 2 \).) At the current level of refinement, advance the data for one timestep, \( \Delta t \), and coarsen the results by a factor of \( n \). On a patch coarsened by a factor of \( n \), coarsen the data from the patch at the current time \( n \) (\( n - 1 \) timesteps, and advance the data for one timestep, \( n \Delta t \). Compare the results of the two time integrations.

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 (since the pseudo-patches are coarsened by a factor of the regrid interval \( n = 2 \), and mesh refinement uses an integer multiple \( r = 2 \) or 4 of the regrid interval). This reduces the work in comparing the errors; in particular, we would not want to increase the work by integrating the fine patches and comparing to the values on a coarser level, since the results on the fine patches would have to be discarded when the patches are moved. The ordering of the coarsening and comparison operations also makes the algorithm simpler: if we were to compare errors on the current level of refinement by coarsening, integrating one coarse step and then refining, we would have to construct a high-order conservative interpolation. In particular, this interpolation would have to be of a higher order than that used to construct predictor values for the numerical flux computation.

It is interesting 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 \( n - 1 \) cells (where \( n \) 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 \( n \) 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.

4.5 Advancing the Data

When we advance the data at a given level of refinement \( \mathcal{L}_f \), we have available to us a grid hierarchy consisting of the current level of refinement and all coarser levels \( \mathcal{L}_{f-1}, \ldots, \mathcal{L}_0 \). We need to perform the steps of operator splitting for the reaction-diffusion system on level \( \mathcal{L}_f \); see section 2.5. This means that we react for a half-step, solve the diffusion equation for a full step, and then react for a half-step until we have completed that number of steps required for synchronization with coarser level \( \mathcal{L}_{f-1} \). 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. Another issue concerns the iterative solution of the linear system for the diffusion; this step involves interesting communication among patches at the same level of refinement and patches at different levels. A third issue involves the relationship of the second reaction to the synchronization of fine data with coarse. We will discuss the reaction steps in this section. The linear algebra will be discussed in section 5.
4.5.1 Initial Operator-Split Reaction

In practical models of electrical wave propagation in the heart [31, 69], there are typically several to
dozens of reactions to be integrated. These may involve significant numerical work, due both to the
nonlinear functions involved and the stiffness of the reactions. As a result, it would be desirable to
concentrate the numerical work for these reactions where the extra work is needed most, namely in
the neighborhood of the reaction front.

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 FitzHugh-Nagumo 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 a coarser level $L_{c-1}$ to determine the initial
data for the linear system on the current level $L_c$. 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.
Since the diffusion is small relative to the reaction in these applications, the difference is negligible.

On the current level $L_c$ of refinement, however, we use the SDIRK scheme to compute the results
after a half-step of reaction. Note that this SDIRK scheme involves a local (at each point in space)
timestepping strategy to guarantee a unique solution to the implicit nonlinear equations. (See section
3.2.5.) These nonlinear equations are uncoupled in space, however, and easy to solve analytically.

4.5.2 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 $L_c$ 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 $L_{c-1}, \ldots, L_0$.

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 diffusion on the current level $L_c$. The exception occurs when the current
level reaches the same time as its coarser level $L_{c-1}$, because it is necessary to synchronize the data
between levels.

We also need to adjust the coarse results near the boundary with the fine grid, in order to avoid
boundary layers. At the end of the last step on the fine level $L_c$, we apply a fine half-step of reaction
to the results of the composite grid diffusion, on all levels $L_{c-1}, \ldots$ that are synchronized with the
current level $L_c$. The results of this reaction replace the previous results from the reaction-diffusion-
reaction step on the coarser level. However, the iterative solution of the diffusion equation on the
composite grid does not provide us with results for the solution on these portions of the coarse level
$L_{c-1}$ overlaying the current level $L_c$. (Recall that the multigrid algorithm computes increments to the
solution on coarser levels.) The coarse grid results in regions overlaying the current level are updated
by a different process.

Because the the fine preference principle, 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 co-incidental 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.

5 Iterative Linear Algebra

During each timestep on each level $\mathcal{L}_0, \ldots, \mathcal{L}_0$ of the hierarchical mesh, we solve a linear system for the Crank-Nicolson discretization of the diffusion equation. The initial data for this diffusion is provided by operator splitting, and is the result of a half-step of reaction. All that remains in the description of the algorithm is to describe how we formulate and solve the linear system for the diffusion equation on the hierarchical grid.

There are several parts to the description of this linear system. First, we will describe the formulation of the linear system on a composite grid, composed of grid cells chosen from the grid hierarchy. These composite grid equations are related to a simple application of a popular technique called mortar finite elements. Then we will describe a conjugate gradient algorithm for this composite grid. As we saw in section 3.3.6, conjugate gradients by itself would involve an order of work that would prevent the resulting algorithm from being competitive with explicit time integration. We obtain a computationally efficient scheme by preconditioning the conjugate gradient iteration with a multiplicative domain decomposition process. If there were no local refinement, this multiplicative domain decomposition would be identical with the multigrid algorithm described in section 3.4.

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. The restriction and prolongation are basically the same as in the multigrid algorithm above, except for special mortar element operations at the interface between coarse and fine grid cells. The smoother operation is performed within a given level of refinement on a list of grid patches; as a result, we have designed a strategy by which the computations in the smoother can be performed in parallel using distributed memory.

5.1 Composite Grid Equations

We begin by describing what we mean by the composite grid when we solve the linear equations for diffusion. At any point in within the computational domain, the grid cell containing that point within the composite grid is chosen from the finest level with a grid patch that contains that point. This is consistent with our Fine Preference assumption for adaptive mesh refinement.

Since we are using a conforming finite element approximation to the diffusion equation, and because of our Grid Alignment assumption, our numerical solution is piecewise linear on the composite grid. This implies that at the interface between coarse and fine grids, the numerical solution on the fine grid is constrained by linear interpolation from the coarse grid. In this sense, the coarse grid is master to the fine grid. As a result, at the interface between coarse and fine grid, there are no fine unknowns in the linear system.

The mortar element approach gives us a useful way to describe the remaining ideas in the composite grid formulation. Suppose that we want to solve the diffusion sub-problem (21) on a domain $\Omega$ that is subdivided into two domains $\Omega_f$ and $\Omega_c$. Let $S$ be the interface between the two subdomains: $S = \overline{\Omega_f} \cap \overline{\Omega_c}$. Using Lagrange multipliers $\mu$ for the normal derivatives at the interface, a weak
formulation of the composite equations is
\[
\frac{d}{dt} \int_{\Omega_f} v_f(x) u_f(x, t) dx + \int_{\Omega_f} \nabla_x v_f(x) \cdot D(x) \nabla_x u_f(x, t) dx - \int_S v_f(x) \mu(x) ds = 0, \forall v_f \in H^1(\Omega_f)
\]  
(35a)
\[
\frac{d}{dt} \int_{\Omega_c} v_c(x) u_c(x, t) dx + \int_{\Omega_c} \nabla_x v_c(x) \cdot D(x) \nabla_x u_c(x, t) dx + \int_S v_c(x) \mu(x) ds = 0, \forall v_c \in H^1(\Omega_c)
\]  
(35b)
\[
\int_S \lambda(-u_j + u_{\ell}) ds = 0, \forall \lambda \in H^{-\frac{1}{2}}(S)
\]  
(35c)

Note that this weak formulation implies that \(u\) and \(n \cdot D \nabla_x u\) are continuous at the interface \(S\). The Crank-Nicolson finite element discretizations of these problems lead to the linear systems
\[
M_f [u_j^{n+1} - u_j^n] + K_f \frac{\Delta t}{2} [u_f^{n+1} + u_f^n] - C_f m = 0
\]  
(36a)
\[
M_c [u_c^{n+1} - u_c^n] + K_c \frac{\Delta t}{2} [u_c^{n+1} + u_c^n] + C_c m = 0
\]  
(36b)
\[
- C_f^T u_j^{n+1} + C_c^T u_c^{n+1} = 0
\]  
(36c)

In these equations, the basis functions for \(\mu\) and \(\lambda\) are piecewise linear on the slave grid, corresponding to \(S \cap \Omega_f\). Further, the entries of \(m\) are the coefficients of the basis functions in the mortar element approximation to \(\mu\).

In one dimension, the mortar \(\mu\) is an unknown number at the point \(S\). Once the integrals have been computed, it is easy to how to write the composite-grid equations near the interface in finite-difference form. If fine grid location \(j\) and coarse grid location \(J\) coincide (i.e., \(x_j = x_J\)), then
\[
\frac{\Delta x_{j-\frac{1}{2}}}{6} [u_{j-1}^{n+1} - u_{j-1}^n] + \frac{\Delta x_{j-\frac{1}{2}}}{3} [u_j^{n+1} - u_j^n] + \frac{D_{j-\frac{1}{2}} \Delta t}{2 \Delta x_{j-\frac{1}{2}}} [u_{j-1}^{n+1} - u_{j-1}^n + u_j^n - u_{j-1}^n] = 0
\]  
(37a)
\[
\frac{\Delta x_{J+\frac{1}{2}}}{3} [u_{J+1}^{n+1} - u_{J+1}^n] + \frac{\Delta x_{J+\frac{1}{2}}}{6} [u_{J+1}^{n+1} - u_{J+1}^n] - \frac{D_{J+\frac{1}{2}} \Delta t}{2 \Delta x_{J+\frac{1}{2}}} [u_{J+1}^{n+1} - u_{J+1}^n + u_{J+1}^n - u_{J-1}^n] = 0
\]  
(37b)
\[
- u_j^{n+1} + u_j^{n+1} = 0
\]  
(37c)

Now we see that it is easy to eliminate variables in this equation and reduce the problem to a self-adjoint system. The final system is the Schur complement formed by using equation (36c) to eliminate the slave unknowns at the interface, and the equation (36a) to define the multipliers. In one dimension, we get
\[
u_j^{n+1} = u_j^{n+1}
\]  
(38a)
\[
\mu = \frac{\Delta x_{j-\frac{1}{2}}}{6} [u_{j-1}^{n+1} - u_{j-1}^n] + \frac{\Delta x_{j-\frac{1}{2}}}{3} [u_j^{n+1} - u_j^n] + \frac{D_{j-\frac{1}{2}} \Delta t}{2} [u_{j-1}^{n+1} - u_{j-1}^n + u_j^n - u_{j-1}^n]
\]  
(38b)
\[
\frac{\Delta x_{J+\frac{1}{2}}}{3} [u_{J+1}^{n+1} - u_{J+1}^n] + \frac{\Delta x_{J+\frac{1}{2}}}{6} [u_{J+1}^{n+1} - u_{J+1}^n] - \frac{D_{J+\frac{1}{2}} \Delta t}{2} [u_{J+1}^{n+1} - u_{J+1}^n + u_{J+1}^n - u_{J-1}^n] = 0
\]  
(38c)
In other words, in one dimension at the interface we finally obtain the composite grid equation

\[
\frac{\Delta x_{j+\frac{1}{2}}}{6}[u_{j+1}^{n+1} - u_j^n] + \frac{\Delta x_{j-\frac{1}{2}}}{3}[u_j^{n+1} - u_j^n] + \frac{D_{j+\frac{1}{2}} \Delta t}{2\Delta x_{j+\frac{1}{2}}}[u_{j+1}^{n+1} - u_j^{n+1} + u_j^n - u_{j-1}^n] \\
+ \frac{\Delta x_{j+\frac{1}{2}}}{3}[u_j^{n+1} - u_j^n] + \frac{\Delta x_{j-\frac{1}{2}}}{6}[u_{j}^{n+1} - u_{j+1}^{n+1}] - \frac{D_{j+\frac{1}{2}} \Delta t}{2\Delta x_{j+\frac{1}{2}}}[u_{j+1}^{n+1} - u_j^{n+1} + u_j^n + u_{j-1}^n - u_j^n] = 0.
\]

This is what we would have obtained if we had applied the finite element method directly to the one-dimensional composite grid.

The advantage of the mortar element formulation is that it helps us organize the communication between levels of refinement, particularly in multiple dimensions. The mortar equations indicate that fine unknowns on the interface between coarse and fine grids must be given the coarse grid value; there is no fine unknown on the coarse-fine interface. In this sense, information flows from the coarse grid to the fine. On the other hand, the mortar (which is the coarse diffusive flux) is computed by summing the fine grid diffusive fluxes and transferring the result to the coarse grid. This communication goes in the reverse direction, and is necessary to maintain a symmetric linear system.

### 5.2 Preconditioned Conjugate Gradients

The preconditioned conjugate gradient algorithm is well-known \([3, 32]\). Given a matrix \(Q \approx A^{-1}\), it performs the following operations:

\[
\begin{align*}
 r &= Ax - b, \quad p = Qr, \quad \gamma = p^T r \\
 &\text{until convergence do} \\
 z &= Ap \\
 \alpha &= \gamma/p^T z, \quad x = x - p\alpha, \quad r + z\alpha \\
 y &= Qr \\
 \delta &= y^T r, \quad p = y + p\delta/\gamma, \quad \gamma = \delta
\end{align*}
\]

(39)

Here \(Ax = b\) is the composite grid linear system for the diffusion equation. The computation of the residual \(r\) involves using the current guesses for the new electrical potentials to compute the diffusive fluxes as in (20), as well as communication between levels to set fine boundary potentials to coarse values and coarse diffusive fluxes (mortars) to fine values. The work in these computations is proportional to the total number of composite grid unknowns.

The computation of \(z\) in the conjugate gradient algorithm is the homogeneous version of the residual computation. The update equations for the solution \(x\), residual \(r\) and search direction \(p\) involve vector-scalar multiplication and vector addition, so this work is proportional to the total number of unknowns. The two inner products involve the same order of work. Thus, it is desirable that the preconditioner \(Q\) involves work proportional to the total number of unknowns as well.

It is known \([3]\) that at each step \(k\), the preconditioned conjugate gradient algorithm minimizes \(r^TA^{-1}r\) over the Krylov subspace \(<r_0, (AQ)r_0, \ldots, (AQ)^k r_0>\). It is also known that the preconditioned conjugate gradient algorithm is equivalent to applying the usual conjugate gradient algorithm to a linear system with matrix \(Q^{\frac{1}{2}}AQ^{\frac{1}{2}}\). It is also interesting to note \([3]\) that the conjugate gradient algorithm minimizes the error over a certain choice of polynomials; in this case,

\[
\|x_{k+1} - x\|_{Q^{\frac{1}{2}}AQ^{\frac{1}{2}}}^2 \equiv (x_{k+1} - x)^T Q^{\frac{1}{2}}AQ^{\frac{1}{2}}(x_{k+1} - x)
\]
\[
\lim_{\pi \in \mathcal{P}} \| (I + Q\Delta AQ^T \pi (Q\Delta AQ^T)) (x_0 - x) \|_{Q\Delta AQ^T}^2 = 0.
\]

This means that if the effect of the preconditioner is that the eigenvalues of \(Q\Delta AQ^T\) are located in a small number of clusters, then there will be a low-order polynomial that nearly annihilates these clusters. The result will be rapid convergence of the preconditioned algorithm.

There is another useful way to view the conjugate gradient iteration and its relationship to the preconditioner. A good preconditioner, such as multigrid, should converge rapidly on its own. It would correspond to setting \(\alpha = 1\) and \(p = y\) in the conjugate gradient algorithm (39) above. For the cost of two inner products, we can modify the multigrid iteration to achieve minimization over expanding Krylov subspaces.

Still, it is reasonable to expect that multigrid should not need extra help. To motivate our use of conjugate gradients, we will consider the one-dimensional linear system for the diffusion equation on a composite grid. Since we use a direct solver on each patch of the composite grid, the only nonzero residuals will occur at the interfaces between levels of refinement. Multigrid will reduce the errors at these remaining locations at the same rate as if we had used a less-effective smoother. However, conjugate gradients will see that the effective number of unknowns in the preconditioned system is roughly the number of interface points, and converge in no more than that number of iterations. Often effective convergence occurs in far fewer iterations than the number of interface points.

There are, of course, advantages to combining conjugate gradients with multigrid in multiple dimensions [1]. We will describe this situation in later papers.

### 5.3 Multiplicative Domain Decomposition

As we mentioned in section 5.2, the conjugate gradient algorithm is preconditioned by a multiplicative domain decomposition iteration. 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 [11, 12].

Again, let us suppose that we have domains \(\Omega_f \subset \Omega_c\). For example, \(\Omega_f\) may correspond to a fine grid in an approximation to the solution of some partial differential equation, and \(\Omega_c\) may correspond to the coarse grid. Note that our Proper Nesting principle implies that all cases within the 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 unknown \(\overline{\mathbf{x}}_f\) and \(\overline{\mathbf{x}}_c\) 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

\[
\begin{bmatrix}
A_{ff} & A_{fc} \\
A_{cf} & A_{cc}
\end{bmatrix}
\begin{bmatrix}
\overline{\mathbf{x}}_f \\
\overline{\mathbf{x}}_c
\end{bmatrix} =
\begin{bmatrix}
b_f \\
b_c
\end{bmatrix},
\]

(40)

**Multiplicative domain decomposition** is a particular iterative improvement algorithm for solving the composite grid equations (40). Without loss of generality, we will assume that our initial guess for the solution of the composite grid equations (40) 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 takes the form

\[
\begin{align*}
\text{compute the fine residual} & \quad r_f^{(c)} = -b_f \\
\text{approximate the solution of the fine equations:} & \quad x_f^{(1)} = -S_{ff}r_f^{(c)} \\
\text{compute the composite grid residual:} & \quad r_c = \begin{bmatrix} r_f^{(1)} \\ r_c^{(0)} \end{bmatrix} = \begin{bmatrix} A_{ff} \\ A_{cf} \end{bmatrix} x_f^{(1)} \\
\text{restrict the fine residual to the coarse grid} & \quad r_{cf} = R r_f \\
\text{approximate the solution of the coarse equations} & \quad x_c = \begin{bmatrix} x_f^{(1)} \\ x_c \end{bmatrix} = -\begin{bmatrix} S_{ff} & S_{fc} \\ S_{cf} & S_{cc} \end{bmatrix} \begin{bmatrix} r_{cf} \\ r_c \end{bmatrix} \\
\text{update the fine grid correction} & \quad x_f^{(2)} = x_f^{(1)} + Px_c \\
\text{compute the fine grid residual} & \quad r_f^{(2)} = r_f^{(0)} + [A_{ff} & A_{fc}] x_f^{(2)} \\
\text{approximate the solution of the fine equations:} & \quad x_f = x_f^{(2)} - S_f r_f^{(2)}
\end{align*}
\]

(41)

Here \( R \) is an inter-grid transfer operator that maps values on the fine grid to values on the coarse grid. Either \( R \) or its adjoint \( R^T \), which maps coarse grid values to fine, should be natural for a given discretization. We choose \( R \) to be the multigrid restriction, described in section 3.4.1, and \( R^T \) to be the prolongation operator. Recall that prolongation is given by injection of the coarse finite element space into the fine space.

In our multiplicative domain decomposition algorithm, the smoother \( S_f \) corresponds to approximating the solution of the differential equation on the subdomain \( \Omega_f \) with appropriate boundary conditions. At physical boundaries, these boundary conditions should be specified. If \( \Omega_f \) is smaller than the physical domain, then it is necessary to select appropriate internal boundary conditions for the definition of \( S_f \). The choices here depend on the problem and the discretization.

Given a hierarchy of nested grids \( \Omega_1 \subset \ldots \subset \Omega_C \), we could perform the multiplicative domain decomposition algorithm recursively between \( \Omega_1 \) and \( \Omega_{1-1} \), \( \Omega_{1-1} \) and \( \Omega_{1-2} \), and so on. In such a case, the approximation of the solution of the coarse equations

\[
\begin{bmatrix} d_{cf} \\ d_c \end{bmatrix} = \begin{bmatrix} S_{ff} & S_{fc} \\ S_{cf} & S_{cc} \end{bmatrix} \begin{bmatrix} r_{cf} \\ r_c \end{bmatrix}
\]

corresponds to recursive application of the algorithm on the coarser grid.

### 5.4 Adaptive Mesh Refinement 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 cannot 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 at some level of refinement may include nodes on the interface between levels and nodes in the interior of the union of the patches on that level. We need to choose a smoother that will perform well within the multiplicative domain decomposition algorithm, and still allow distributed computation.

Here a Jacobi iteration is useful. 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. At nodes on the interface with a coarser grid, a boolean mask tells us to ignore the correction from the Jacobi iteration. We will describe this approach in a later paper.

6 Numerical Results

In order to illustrate the convergence of the multiplicative domain decomposition, we have performed simulations with very small tolerances for the convergence of the iterative linear algebra. Our convergence criterion is that the maximum absolute value of the residual should be small relative to the maximum absolute value of the diffusive fluxes, and relative to the maximum absolute of the inhomogeneities in the linear equations. Figure ?? shows the convergence of the multiplicative domain decomposition iteration for the first and last linear systems a the simulation with 3 levels in adaptive mesh refinement. The iteration converges very rapidly initially, when the solution of the system is essentially piecewise constant; in this case, only 5 iterations were needed to obtain a residual reduced by roughly 12 orders of magnitude. At later times the iteration is still linearly convergent, but at a slower rate; in this case, 11 iterations were needed. In practical calculations, only one or two multiplicative domain decomposition iterations are needed to reduce the residual by a factor of $10^{-3}$. Given that we seldom resolve the front speed so accurately, this is a sufficient number of iterations for most simulation.

Figure 5: Convergence of Multiplicative Domain Decomposition Diamonds: iteration in first step of a simulation with 3 levels; Circles: iteration in last step of the same simulation
7 Acknowledgements

The authors would like to thank Thomas Witelski for several useful conservations during the preparation of this paper.

References


