Annals of Biomedical Engineering, Vol. 36, No. 8, August 2008 (© *2008*) pp. 1398–1408
DOI: 10.1007/s10439-008-9513-0

An Efficient Technique for the Numerical Solution of the Bidomain Equations

JONATHAN P. WHITELEY

Oxford University Computing Laboratory, Wolfson Building, Parks Road, Oxford OX1 3QD, UK

(*Received 8 January 2008; accepted 1 May 2008; published online 15 May 2008*)

Abstract—Computing the numerical solution of the bidomain equations is widely accepted to be a significant computational challenge. In this study we extend a previously published semi-implicit numerical scheme with good stability properties that has been used to solve the bidomain equations (Whiteley, J.P. IEEE Trans. Biomed. Eng. 53:2139–2147, 2006). A new, efficient numerical scheme is developed which utilizes the observation that the only component of the ionic current that must be calculated on a fine spatial mesh and updated frequently is the fast sodium current. Other components of the ionic current may be calculated on a coarser mesh and updated less frequently, and then interpolated onto the finer mesh. Use of this technique to calculate the transmembrane potential and extracellular potential induces very little error in the solution. For the simulations presented in this study an increase in computational efficiency of over two orders of magnitude over standard numerical techniques is obtained.

Keywords—Cardiac electrophysiological modeling, Stability of numerical schemes, Efficient numerical schemes.

INTRODUCTION

Cardiac electrophysiology in tissue is usually modeled using the bidomain equations: see, for example, Keener and Sneyd[9] for a derivation of these equations. These equations consist of an elliptic partial differential equation and a parabolic partial differential equation, coupled at each point in space with a large system of stiff, non-linear ordinary differential equations. The actual size of the system of ordinary differential equations depends on the electrophysiological model used—see Nickerson[11] for a collection of these models. It is widely accepted that the accurate solution of these equations in a three-dimensional computational domain that represents a mammalian heart is a significant computational challenge, both in terms of computation time and memory required.

One cause of the large computation time required to solve the bidomain equations numerically is the stiffness of the ordinary differential equations and parabolic partial differential equation. Explicit numerical methods are often inefficient when used to solve stiff ordinary differential equations, as the maximum timestep that may be used is usually dictated by stability considerations, and not by the timescales that the differential equations model. These stability issues may be avoided by the use of an implicit numerical method.[7] However, when solving the bidomain equations, the use of an implicit numerical method for both the ordinary differential equations and the parabolic partial differential equation results in a very large coupled non-linear system that much be solved on each timestep—solving this system requires that a large Jacobian matrix must be computed and stored in the computer's memory. Furthermore, due to the complex nature of the terms appearing in the differential equations, writing software to calculate the Jacobian matrix is an error-prone activity. To the best of our knowledge, this approach has only been applied when the very simple Fitzhugh–Nagumo electrophysiological model[10] is used. In this special case only one ordinary differential equation is coupled to the parabolic and elliptic partial differential equations at each point in space. Due to the small size of the system of ordinary differential equations, the problems highlighted above when using a fully implicit numerical solver are not too great.

Under some circumstances the elliptic partial differential equation may be eliminated from the bidomain equations—the resulting system of equations is known as the monodomain equations. Various techniques have been used to reduce the computational effort required to solve the bidomain equations or monodomain equations when a more complex electrophysiological model than the Fitzhugh–Nagumo model is used. For example: operator splitting techniques[6,17,20–22]; semi-implicit numerical

Address correspondence to Jonathan P. Whiteley, Oxford University Computing Laboratory, Wolfson Building, Parks Road, Oxford OX1 3QD, UK. Electronic mail: Jonathan.Whiteley@comlab.ox.ac.uk

discretizations[4,8,24,28,29]; a judicious choice of linear algebra solver and preconditioner for solving the linear systems that arise from the discretization of the differential equations[14,15,27]; domain decomposition techniques[18]; adaptive numerical techniques[3,13,17,29]; taking account of fibre orientation to render the linear system arising more tractable[26]; predictor–corrector techniques[19]; and artificially slowing down the faster physiological processes that occur so that they may be computed using a longer timestep.[1] Although these techniques do reduce computation time, the solution of the system of equations on a realistic sized computational domain still poses a significant challenge.

The processes modeled by the electrophysiological model occur on a wide variety of time scales, which has the effect that they are also observed on a wide range of length scales. This observation has previously been utilized by other authors[3,18,29] when developing an adaptive algorithm for the solution of the bidomain equations. The algorithm proposed by Whiteley[29] demonstrated that an increase in computational efficiency of over two orders of magnitude can be achieved when the fast processes—i.e., those processes that cause the action potential upstroke—are isolated to a small portion of the time simulated. This assumption is, however, not always true: it is true for a normal heartbeat, but is not true for a fibrillating heart. In this study we develop a numerical technique that is slightly more efficient than the previous algorithm described by Whiteley,[29] and with applicability to a much wider range of problems. Although we describe the application of this technique to the bidomain equations, the technique may be applied equally effectively to the monodomain equations.

THE NUMERICAL METHOD

In this section we describe a new, efficient algorithm that may be used to solve the bidomain equations numerically. We begin by writing down the bidomain equations, and the semi-implicit algorithm on which the new algorithm is based. We then describe the new algorithm.

The Differential Equations

The bidomain equations are given by:

$$\chi\left(C_{\mathrm{m}}\frac{\partial V_{\mathrm{m}}}{\partial t}+I_{\mathrm{ion}}(\mathbf{u},V_{\mathrm{m}})\right)-\nabla\cdot(\sigma_{\mathrm{i}}\nabla(V_{\mathrm{m}}+\phi_{\mathrm{e}}))=I_{\mathrm{s_i}}, \tag{1}$$

$$\nabla\cdot((\sigma_{\mathrm{i}}+\sigma_{\mathrm{e}})\nabla\phi_{\mathrm{e}}+\sigma_{\mathrm{i}}\nabla V_{\mathrm{m}})=I_{\mathrm{s_e}}, \tag{2}$$

$$\frac{\partial \mathbf{u}}{\partial t}=\mathbf{f}(\mathbf{u},V_{\mathrm{m}}), \tag{3}$$

where $V_{\mathrm{m}}(\mathbf{x}, t)$ is the transmembrane potential, $\phi_{\mathrm{e}}(\mathbf{x}, t)$ is the extracellular potential, χ is the surface to volume ratio, C_{m} is the membrane capacitance per unit area, σ_{i} is the intracellular conductivity tensor, σ_{e} is the extracellular conductivity tensor, I_{ion} is the ionic current, $I_{\mathrm{s_i}}$ is the external stimulus applied to the intracellular space, $I_{\mathrm{s_e}}$ is the external stimulus applied to the extracellular space, $\mathbf{u}(\mathbf{x}, t)$ is a vector containing gating variables and concentrations, $\mathbf{f}$ is a prescribed vector-valued function, $\mathbf{x}$ is position, and t is time. Both I_{ion} and the function $\mathbf{f}$ are determined by the electrophysiological model used.

Equations (1) and (2) include spatial derivatives, and therefore require boundary conditions. We assume no current crosses the boundary. Noting that the intracellular potential ϕ_{i} is given by $\phi_{\mathrm{i}} = V_{\mathrm{m}} + \phi_{\mathrm{e}}$, the intracellular and extracellular currents, i_{i} and i_{e}, are given by

$$i_{\mathrm{i}}=-\sigma_{\mathrm{i}}\nabla(V_{\mathrm{m}}+\phi_{\mathrm{e}}),\quad i_{\mathrm{e}}=-\sigma_{\mathrm{e}}\nabla\phi_{\mathrm{e}}. \tag{4}$$

Boundary conditions that enforce no flow of current across the boundary are therefore given by

$$\mathbf{n}\cdot(\sigma_{\mathrm{i}}\nabla(V_{\mathrm{m}}+\phi_{\mathrm{e}}))=0,\quad \mathbf{n}\cdot(\sigma_{\mathrm{e}}\nabla\phi_{\mathrm{e}})=0, \tag{5}$$

where $\mathbf{n}$ is the outward pointing unit normal vector to the computational domain.

We note that Eqs. (1)–(3) together with boundary conditions given by Eq. (5) do not have a unique solution as ϕ_{e} is only defined up to an additive constant. This non-uniqueness does not affect the physiology modeled by the bidomain equations as in Eq. (4) the extracellular current is determined by derivatives of ϕ_{e} and is therefore independent of the additive constant. This additive constant may be determined by posing Eqs. (1) and (2) in terms of ϕ_{i} and ϕ_{e}. These equations[9] contain time derivatives as well as spatial derivatives of ϕ_{i} and ϕ_{e}, and so the solution does not contain any arbitrary constant.

The Basis of the Technique

The algorithm developed here is based on the semi-implicit technique previously used by Whiteley.[28,29] The dependent variables V_{m}, ϕ_{e}, and $\mathbf{u}$ are calculated at each of the discrete times $t_0, t_1, t_2, \ldots, t_N$. We use the notation

$$V_{\mathrm{m}}^n(\mathbf{x})=V_{\mathrm{m}}(\mathbf{x},t_n),\quad \phi_{\mathrm{e}}^n(\mathbf{x})=\phi_{\mathrm{e}}(\mathbf{x},t_n),$$
$$\mathbf{u}^n(\mathbf{x})=\mathbf{u}(\mathbf{x},t_n),$$

for these quantities. On each timestep we begin by solving Eqs. (1) and (2), with the conduction terms being treated implicitly, and the biochemical reaction terms being treated explicitly. This leads to the following system of equations:

$$\frac{\chi C_{\mathrm{m}}}{\Delta t_n} V_{\mathrm{m}}^n - \nabla \cdot \left(\sigma_{\mathrm{i}} \nabla \left(V_{\mathrm{m}}^n + \phi_{\mathrm{e}}^n\right)\right) = \frac{\chi C_{\mathrm{m}}}{\Delta t_n} V_{\mathrm{m}}^{n-1} + I_{\mathrm{s_i}} - \chi I_{\mathrm{ion}}\left(V_{\mathrm{m}}^{n-1}, \mathbf{u}^{n-1}\right), \qquad (6)$$

$$\nabla \cdot \left((\sigma_{\mathrm{i}} + \sigma_{\mathrm{e}}) \nabla \phi_{\mathrm{e}}^n + \sigma_{\mathrm{i}} \nabla V_{\mathrm{m}}^n\right) = I_{\mathrm{s_e}}, \qquad (7)$$

where $\Delta t_n = t_n - t_{n-1}$. Equations (6) and (7) are usually discretized in space using either the finite element method or the finite difference method, allowing the calculation of V_{m}^n and ϕ_{e}^n. For either of these methods, the discretization results in the matrix equation

$$A \begin{pmatrix} \mathbf{V_m^n} \\ \boldsymbol{\phi}_\mathbf{e}^\mathbf{n} \end{pmatrix} = \begin{pmatrix} \mathbf{b_V} \\ \mathbf{b_e} \end{pmatrix}, \qquad (8)$$

where A is the matrix arising from the finite element or finite difference discretization in space of Eqs. (6) and (7), $\mathbf{V_m^n}$ are the unknowns associated with the discretization of $V_m^n(\mathbf{x})$, $\boldsymbol{\phi}_\mathbf{e}^\mathbf{n}$ are the unknowns associated with the discretization of $\phi_{\mathrm{e}}^n(\mathbf{x})$, $\mathbf{b_V}$ arises from the right-hand-side of Eq. (6), and $\mathbf{b_e}$ arises from the right-hand-side of Eq. (7). Having solved this matrix equation to calculate V_{m}^n and ϕ_{e}^n, we then solve the ordinary differential equations given by Eq. (3) at each point in space to calculate $\mathbf{u}^n$. In common with previous work[28,29] we use the backward Euler method to solve these ordinary differential equations.

We noted in section "The Differential Equations" that ϕ_{e} was only determined up to an additive constant, and so we may add any constant value to each of the entries of $\boldsymbol{\phi}_\mathbf{e}^\mathbf{n}$. This has the effect that the matrix A in Eq. (8) is singular, with a nullspace that is spanned by the vector

$$\begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ \vdots \\ 1 \end{pmatrix},$$

where the entries corresponding to the vector $\mathbf{V_m^n}$ take the value zero, and the entries corresponding to the vector $\boldsymbol{\phi}_\mathbf{e}^\mathbf{n}$ take the value 1. As the linear system given by Eq. (8) is symmetric, the nullspace of the matrix A is identical to the nullspace of the transpose of A. Under these conditions the iterative solver GMRES converges to the solution of Eq. (8) without requiring any constraint to make A non-singular.[2] This solution technique only determines values of entries of $\boldsymbol{\phi}_\mathbf{e}^\mathbf{n}$ up to an additive constant, as discussed in section "The Differential Equations". The value of the additive constant will be different on each timestep, and is dependent on both the preconditioner used and the initial solution guess. An alternative strategy would be to fix one value of ϕ_{e} throughout the simulation.

The Efficient Algorithm

When using the algorithm described in section "The Basis of the Technique", the matrix A appearing in Eq. (8) is the same on each timestep and therefore need only be calculated once. The bulk of the computational effort required on each timestep therefore consists of: (i) calculating the right-hand-side of Eq. (8); (ii) solving the matrix equation given by Eq. (8); and (iii) solving the ordinary differential equations given by Eq. (3). The algorithm proposed here substantially reduces the computational effort associated with the first and third of these tasks, and is based on the observation that very few of the processes modeled by electrophysiological models occur on shorter time scales and shorter length scales. We now explain how to exploit this observation.

We begin by solving the bidomain equations, using the electrophysiological model described by Noble *et al.*[12] on a square with sides 20 mm using the algorithm described by Whiteley.[28] One corner of this domain was stimulated to generate an action potential propagating across the square. The total ionic current at the point at the center of the square is plotted against time in Fig. 1a. We see that the total ionic current takes its largest magnitude at around 0.0521 s, and that the magnitude of this current is much smaller elsewhere. The large magnitude of the total ionic current seen in Fig. 1a is mainly due to the fast sodium current: in Fig. 1b we plot the total ionic current with the fast sodium current removed. We see that the magnitude of this current is much smaller than the magnitude of the total ionic current. Similar features of the total ionic current and the fast sodium current are seen in space. In Fig. 1c we plot the total ionic current at the cross-section $y = 10$ mm across the square at time 0.0521 s. The total current with the fast sodium current removed is plotted in Fig. 1d. We see that the fast sodium current is again responsible for the large magnitude of the total ionic current.

As the ionic current drives changes in V_{m} and ϕ_{e} in Eqs. (1) and (2), any large current must be accurately calculated. It therefore seems appropriate to calculate the fast sodium current on a much finer spatial mesh and to update this current more frequently than the other components of the ionic current. An example of a finer spatial mesh that may be used is shown in Fig. 2. When using the meshes shown in this figure the fast sodium current is calculated using the mesh consisting of fine lines, with nodal spacing given by h_{fast}.

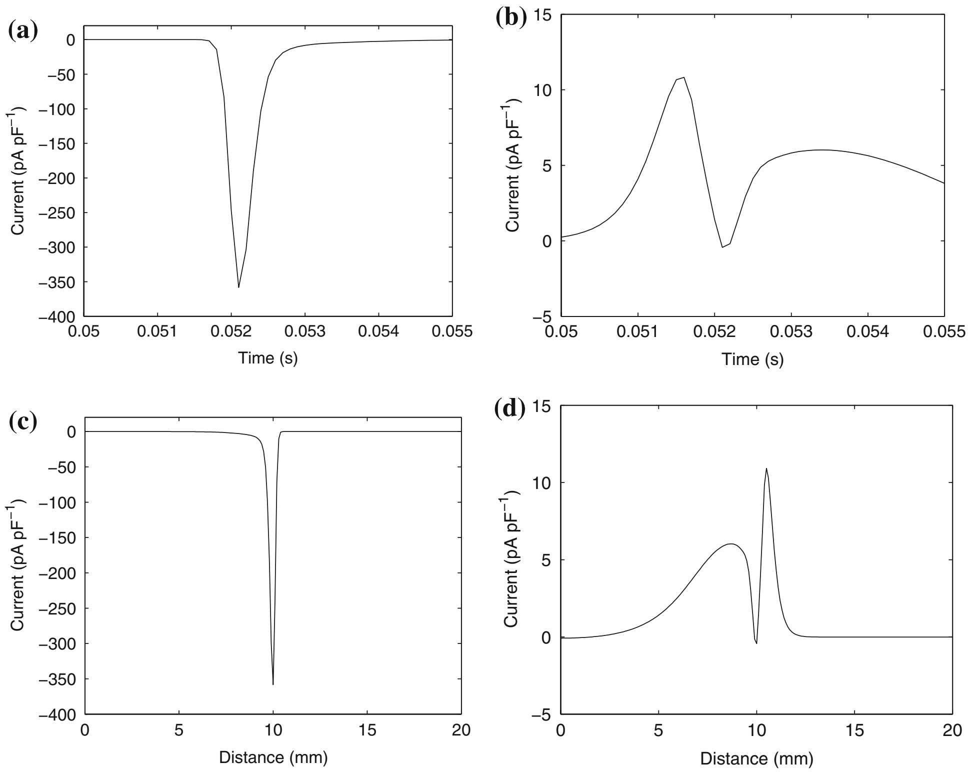

FIGURE 1. (a) The total ionic current as a function of time; (b) The total ionic current with the fast sodium current removed as a function of time; (c) The total ionic current as a function of space; (d) The total ionic current with the fast sodium current removed as a function of space. See text for more details of the simulation.

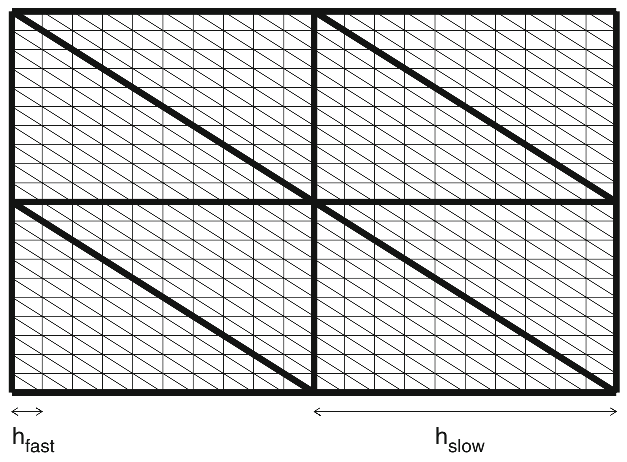

FIGURE 2. The fine mesh (thin lines) with nodal spacing h_{fast}, and the coarse mesh (thick lines) with nodal spacing h_{slow}.

The other contributions to the ionic current are calculated on the mesh consisting of thicker lines, with nodal spacing given by h_{slow}. As well as defining h_{fast} and h_{slow}, we will also need one timestep on which the fast currents are updated, Δt_{fast}, and another timestep on which the slower currents are updated, Δt_{slow}.

We now write Eqs. (6) and (7) as

$$\frac{\chi C_{\text{m}}}{\Delta t_n} V_{\text{m}}^n - \nabla \cdot \left(\sigma_{\text{i}} \nabla \left(V_{\text{m}}^n + \phi_{\text{e}}^n\right)\right) = \Gamma_1 + \Gamma_2, \qquad (9)$$

$$\nabla \cdot \left((\sigma_{\text{i}} + \sigma_{\text{e}}) \nabla \phi_{\text{e}}^n + \sigma_{\text{i}} \nabla V_{\text{m}}^n\right) = \Gamma_3, \qquad (10)$$

where, denoting the fast sodium current by I_{Na},

$$\Gamma_1 = \frac{\chi C_{\text{m}}}{\Delta t_n} V_{\text{m}}^{n-1} + I_{\text{s}_{\text{i}}} - \chi I_{\text{Na}},$$
$$\Gamma_2 = -\chi(I_{\text{ion}} - I_{\text{Na}}),$$
$$\Gamma_3 = I_{\text{s}_{\text{e}}}.$$

By writing Eqs. (6) and (7) in this way, we have combined the fast variables—V_{m}^{n-1}, I_{Na}, $I_{\text{s}_{\text{i}}}$ and $I_{\text{s}_{\text{e}}}$—into terms that are separate from the slower variables. This allows us to write Eq. (8) as

$$A \begin{pmatrix} \mathbf{V}_{\mathbf{m}}^{\mathbf{n}} \\ \boldsymbol{\phi}_{\mathbf{e}}^{\mathbf{n}} \end{pmatrix} = \begin{pmatrix} \mathbf{b}_{\mathbf{V}}^{\text{fast}} \\ \mathbf{b}_{\mathbf{e}}^{\text{fast}} \end{pmatrix} + \begin{pmatrix} \mathbf{b}_{\mathbf{V}}^{\text{slow}} \\ \mathbf{0} \end{pmatrix}, \qquad (11)$$

where $\mathbf{b}_{\mathbf{V}}^{\text{fast}}$ arises from Γ_1, $\mathbf{b}_{\mathbf{V}}^{\text{slow}}$ arises from Γ_2, and $\mathbf{b}_{\mathbf{e}}^{\text{fast}}$ arises from Γ_3.

We are now in a position to write down the new algorithm. When computing the right-hand-side of Eq. (11), the vectors $\mathbf{b}_{\mathbf{V}}^{\text{fast}}$ and $\mathbf{b}_{\mathbf{e}}^{\text{fast}}$ are calculated in the usual way using the fine mesh shown in Fig. 2. The vector $\mathbf{b}_V^{\text{slow}}$ is calculated by first calculating the quantities required at the nodes of the coarse mesh shown in Fig. 2. Interpolation is then used to calculate these quantities on the fine mesh, allowing the whole of the right-hand-side of Eq. (11) to be computed on the fine mesh. A timestep Δt_{fast} is used when solving Eq. (11). However, as the quantities contained in $\mathbf{b}_V^{\text{slow}}$ vary on a slower time scale, this vector is not updated on each timestep. Instead, it is updated using a longer timestep Δt_{slow}. The method described in this paragraph allows us to update V_m and ϕ_e using a timestep Δt_{fast}.

Having described our technique for solving Eqs. (6) and (7) numerically to calculate V_m^n and ϕ_e^n we now turn our attention to solving the ordinary differential equations given by Eq. (3). As the components of $\mathbf{b}_V^{\text{slow}}$ are calculated using only quantities at the nodes of the coarse mesh, we only calculate the solution of most components of the solution of Eq. (3) at these points—this allows a significant saving in terms of both computational time and memory required. The only components of the solution of Eq. (3) that must be calculated at the nodes of the fine mesh are those that are needed when calculating the fast sodium current. When using the electrophysiological model described by Noble *et al.*[12] the fast sodium current is given by

$$I_{\text{Na}} = g_{\text{Na}} m^3 h (V_m - E_{\text{Na}}),$$
$$\text{where} \quad E_{\text{Na}} = \frac{RT}{F} \log \frac{\text{Na}_o + P_{\text{nak}} \text{K}_o}{\text{Na}_i + P_{\text{nak}} \text{K}_i}.$$

All terms in this expression—with the exception of the gating variables m, h, the transmembrane potential V_m, the intracellular sodium concentration Na_i and the intracellular potassium concentration K_i—are given by constants. V_m is calculated from the solution of Eq. (11) at each point of the fine mesh. The quantities Na_i and K_i only affect I_{Na} through the term E_{Na}. In Fig. 3 we plot E_{Na} at the point ($x = 10$ mm, $y = 10$ mm) during the simulation described earlier in this section. We see that this quantity does not vary significantly during the simulation and so it may be calculated by approximating Na_i and K_i on the coarse mesh and interpolating these quantities onto the fine mesh. The gating variables m and h do, however, vary rapidly and should be calculated at each node of the fine mesh. Fortunately this is not too computationally expensive as these variables satisfy differential equations of the form

$$\frac{\partial m}{\partial t} = a_m(V_m) + b_m(V_m)m,$$
$$\frac{\partial h}{\partial t} = a_h(V_m) + b_h(V_m)h,$$

for suitable functions a_m, b_m, a_h, b_h. These equations are linear in m and h, and so a wide variety of techniques may be used to efficiently integrate these equations numerically.

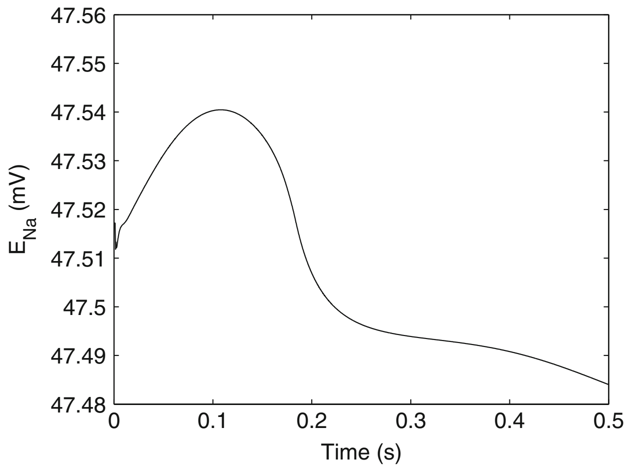

FIGURE 3. The quantity E_{Na} as a function of time at the point (x = 10 mm, y = 10 mm).

Solving the ordinary differential equations for most variables on a coarser mesh allows a significant gain in computational efficiency. Although the equations that are solved on the coarse mesh do not result in large, rapidly varying contributions to the ionic transmembrane current, it is still possible that these equations may model processes that vary rapidly. As such, it is not possible to claim that these equations should be solved using a longer timestep and so we use a timestep Δt_{fast} for the numerical solution of all ordinary differential equations.

This completes the description of the algorithm. We now turn our attention to assessing the accuracy and efficiency of this algorithm.

NUMERICAL EXPERIMENTS

In this section we investigate the performance of the algorithm described in section "The Efficient Algorithm". We describe the simulations that are performed before discussing first the accuracy and then the computational efficiency of the algorithm.

Summary of Computations

All the simulations were carried out on a square occupying the region $0 < x, \ y < 20$ mm with fibres parallel to the x-axis. The electrophysiological model used is that described by Noble *et al.*[12] Other parameter values used are identical to those used previously[28] with the exception of the conductivity tensors. The intracellular conductivities were 0.13 mS mm^{-1} along the fibre and 0.026 mS mm^{-1} perpendicular to the fibre, while the extracellular conductivities were 0.13 mS mm^{-1} along the fibre and 0.065 mS mm^{-1} perpendicular to the fibre. One corner of the square was stimulated at time 0.001 s, generating an action potential that propagated across the square.

For each simulation the transmembrane potential was recorded at the points ($x = 10$ mm, $y = 10$ mm), ($x = 20$ mm, $y = 10$ mm) and ($x = 10$ mm, $y = 20$ mm). This allowed us to calculate: (i) v_F, the average conduction velocity in the direction of the fibres between the points ($x = 10$ mm, $y = 10$ mm), ($x = 20$ mm, $y = 10$ mm); (ii) v_{PF}, the average conduction velocity in the direction perpendicular to the fibres between the points ($x = 10$ mm, $y = 10$ mm), ($x = 10$ mm, $y = 20$ mm); and (iii) the maximum slope of the action potential upstroke at the point ($x = 10$ mm, $y = 10$ mm). The activation time, $A(x,y)$, at each node of the fine mesh—defined to be when V_m first took the value −85 mV was also recorded. The normal velocity of the action potential wavefront, v_N, is then given by[23]

$$v_N = \frac{1}{|\nabla A|}.$$

We record the value of v_N at the point ($x = 10$ mm, $y = 10$ mm) by calculating this quantity in the element with bottom right hand corner at this point. The Action Potential Duration (APD)—defined to be the time for which $V_m > -85$ mV—was also recorded at each node of the fine mesh.

To investigate the accuracy of ϕ_e calculated using the proposed algorithm we calculate the magnitude of the extracellular current, given by $|\sigma_e \nabla \phi_e|$, at the point ($x = 10$ mm, $y = 10$ mm).

In previous work[29] we have shown that if a uniform spatial mesh and timestep are used with the electrophysiological model used in this study,[12] then we should use a spatial nodal spacing of around 0.1 mm, and a timestep of around 10^{-4} s. As the parameters h_{fast} and Δt_{fast} are chosen to capture the fast physiological processes occurring, in all simulations presented here we use the values $h_{fast} = 0.1$ mm and $\Delta t_{fast} = 10^{-4}$ s. Suitable values of h_{slow} and Δt_{slow} are to be determined from our simulations. We investigate the effect on accuracy and computation time of choosing $h_{slow} =$ 0.1 mm, 0.5 mm, 1.0 mm, and on choosing $\Delta t_{slow} = 10^{-4}$ s and 10^{-3} s.

The computation time required for each of the simulations is recorded. This allows the effect on computational efficiency of the choice of h_{slow} and Δt_{slow} to be evaluated. Further simulations are carried out using the adaptive algorithm described by Whiteley[29] to compare the efficiency of the new algorithm with this previously published adaptive algorithm. Unless otherwise stated a time period of 0.35 s was simulated, as this allows the action potential to be completed at all points of the square.

In this study, the linear systems were solved using the ILU preconditioned GMRES routines provided by the PETSc libraries.[16] Lookup tables[5,25] were used to calculate as many of the complex non-linear terms as possible.

Accuracy of the Algorithm

We begin by investigating the effect of Δt_{slow} on the action potential at the point ($x = 10$ mm, $y = 10$ mm). In Fig. 4a we plot the action potential recorded at this point using two different values of Δt_{slow} : 10^{-4} s (solid line); and 10^{-3} s (broken line). In both cases $h_{slow} = h_{fast}$. We see that both action potentials are almost visually indistinguishable from each other—only when the upstroke of these action potentials is plotted using an expanded time axis in Fig. 4b do we see a difference between the two action potentials. For both action potentials plotted in Fig. 4 the action potential duration was 0.179 s.

We now turn our attention to investigating the effect of h_{slow} on the action potential at the point ($x = 10$ mm,

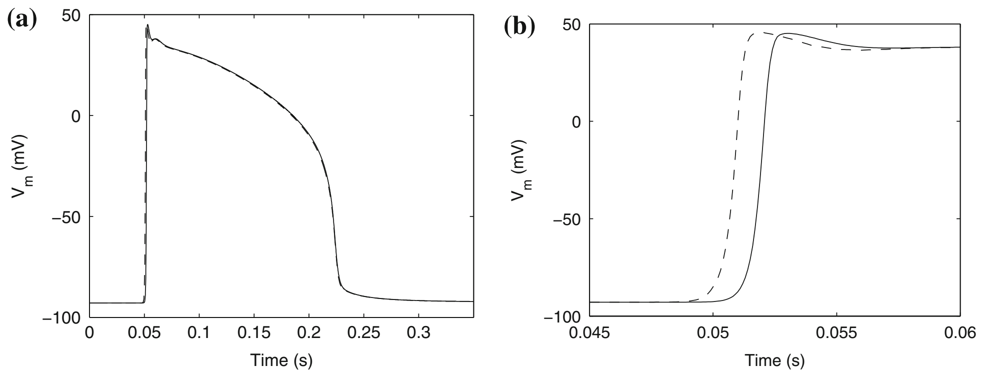

FIGURE 4. The action potential at the point ($x = 10$ mm, $y = 10$ mm) calculated when $h_{slow} = h_{fast}$, and $\Delta t_{slow} = 10^{-4}$ s (solid line) and $\Delta t_{slow} = 10^{-3}$ s (broken line). (a) shows the whole action potential; (b) shows the upstroke of the action potentials on an expanded time axis.

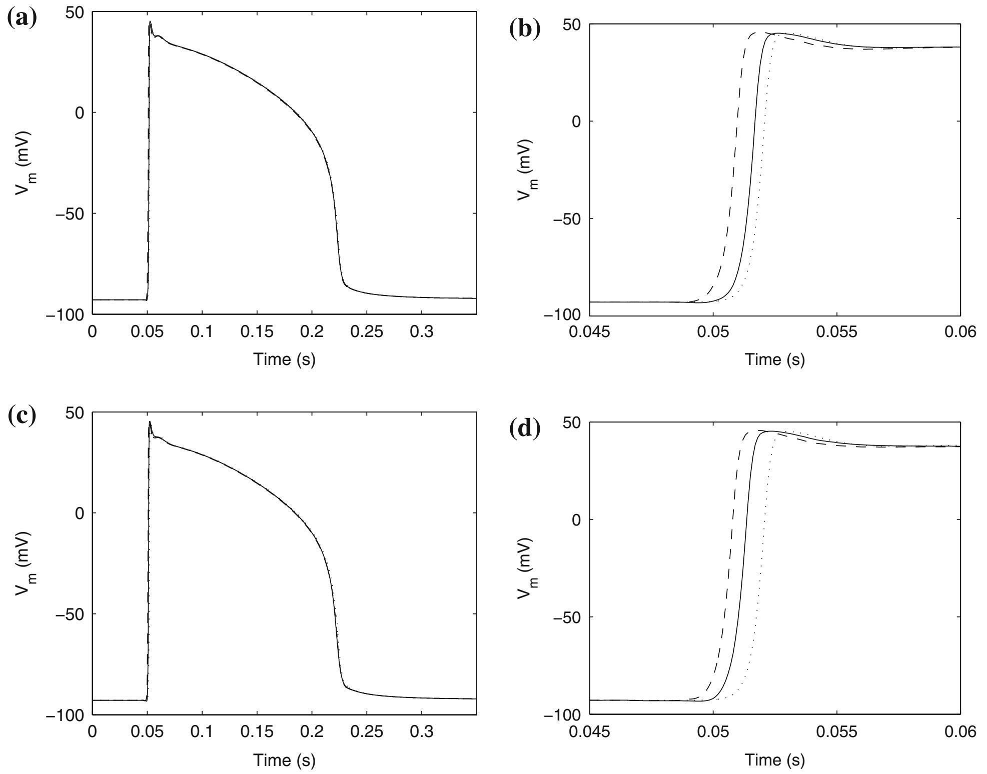

FIGURE 5. The action potential at the point (x = 10 mm, y = 10 mm). In all plots the dotted line represents the simulation with $h_{slow} = h_{fast}$, and $\Delta t_{slow} = \Delta t_{fast}$. In (a) the solid line represents the simulation with h_{slow} = 0.5 mm and $\Delta t_{slow} = 10^{-4}$ s, and the broken line represents the simulation with h_{slow} = 0.5 mm and $\Delta t_{slow} = 10^{-3}$; (b) plots the action potentials seen in (a) on an expanded time axis. In (c) the solid line represents the simulation with h_{slow} = 1.0 mm and $\Delta t_{slow} = 10^{-4}$ s, and the broken line represents the simulation with h_{slow} = 1.0 mm and $\Delta t_{slow} = 10^{-3}$; (d) plots the action potentials seen in (c) on an expanded time axis.

y = 10 mm). In Figs. 5a and 5b we plot the action potential recorded at this point using h_{slow} = 0.5 mm and $\Delta t_{slow} = 10^{-4}$ s (solid line) and h_{slow} = 0.5 mm and $\Delta t_{slow} = 10^{-3}$ s (broken line). In Figs. 5c and 5d we plot the action potential calculated using h_{slow} = 1.0 mm and $\Delta t_{slow} = 10^{-4}$ s (solid line) and h_{slow} = 1.0 mm and $\Delta t_{slow} = 10^{-3}$ s (broken line). In every plot in Fig. 5 the dotted line represents the action potential calculated using $h_{slow} = h_{fast}$ and $\Delta t_{slow} = \Delta t_{fast}$. In all cases, the action potentials are almost visually indistinguishable from each other except in Figs. 5b and 5d where the upstrokes of these action potentials have been plotted on an expanded time axis. For all action potentials plotted in Fig. 5 the action potential duration was 0.179 s.

In Table 1 we tabulate the quantities v_F, v_{PF}, v_N, and the maximum gradient of the upstroke described in section "Summary of Computations" as a function of h_{slow} and Δt_{slow}. We note that v_N is smaller than the magnitude of the vector (v_F, v_{PF}): this is because v_F and v_{PF} are average velocities while v_N is the velocity at a point of the tissue. For all values of h_{slow} and Δt_{slow} used here the difference between the simulation with $h_{slow} = h_{fast}$ and $\Delta t_{slow} = \Delta t_{fast}$ is small: the maximum difference in v_F is 3.9%; the maximum difference in v_{PF} is 2.7%; the maximum difference in v_N is 11.5%; and the maximum difference in the maximum gradient of the upstroke is 1.0%. For each individual simulation described in this paper, the average difference across the whole computational domain between v_N and the value of v_N calculated using $h_{slow} = h_{fast}$ and $\Delta t_{slow} = \Delta t_{fast}$ was less than 8%.

TABLE 1. The quantities v_F, v_{PF}, v_N and maximum upstroke derivative calculated at the point (x = 10 mm, y = 10 mm).

h_{slow} (mm)	Δt_{slow} (s)	v_F (mm s^{-1})	v_{PF} (mm s^{-1})	v_N (mm s^{-1})	max $\frac{dV_m}{dt}$ (mV s^{-1})
0.1	10^{-4}	718.9	243.2	246.7	1.737
0.5	10^{-4}	717.7	245.8	236.6	1.736
1.0	10^{-4}	727.5	244.0	238.7	1.721
0.1	10^{-3}	730.1	248.5	231.9	1.741
0.5	10^{-3}	724.7	249.8	218.3	1.720
1.0	10^{-3}	747.0	246.3	249.0	1.732

The APD calculated using $h_{slow} = h_{fast}$ and $\Delta t_{slow} = \Delta t_{fast}$ varied across the domain from 0.176 s to 0.185 s. The maximum difference between the simulation with $h_{slow} = h_{fast}$ and $\Delta t_{slow} = \Delta t_{fast}$ and simulations using other values of h_{slow} and Δt_{slow} are listed in Table 2. We see that the error in calculating the APD that is induced by using a coarse mesh is also small, the relative error always being less than 2.1%.

To investigate the effect of Δt_{slow} on ϕ_e we plot the magnitude of the extracellular current at the point ($x = 10$ mm, $y = 10$ mm) in Fig. 6a. In both simulations we use $h_{\text{slow}} = h_{\text{fast}}$. The solid line represents a simulation with $\Delta t_{\text{slow}} = 10^{-4}$ s while the broken line represents a simulation with $\Delta t_{\text{slow}} = 10^{-3}$ s. Figure 6b shows the same plots, but with the rapidly varying features shown on an expanded timescale. We see that difference between the magnitude of the extracellular current calculated in the two simulations are acceptably small for almost all purposes. Similar observations are made when investigating the effect of h_{slow} on the magnitude of the extracellular current. In Figs. 7a and 7b we plot the magnitude of the extracellular current at this point using $h_{\text{slow}} = 0.5$ mm and $\Delta t_{\text{slow}} = 10^{-4}$ s (solid line) and $h_{\text{slow}} = 0.5$ mm and $\Delta t_{\text{slow}} = 10^{-3}$ s (broken line). In Figs. 5c and 5d we plot the magnitude of the extracellular current calculated using $h_{\text{slow}} = 1.0$ mm and $\Delta t_{\text{slow}} = 10^{-4}$ s (solid line) and $h_{\text{slow}} = 1.0$ mm and $\Delta t_{\text{slow}} = 10^{-3}$ s (broken line). In every plot in Fig. 7 the dotted line represents the magnitude of the extracellular current calculated using $h_{\text{slow}} = h_{\text{fast}}$ and $\Delta t_{\text{slow}} = \Delta t_{\text{fast}}$. We see that the magnitude of the extracellular current is captured accurately by all values of Δt_{slow} and h_{slow} used in this study.

TABLE 2. Maximum relative difference and absolute difference in APD calculated using $h_{\text{slow}} = h_{\text{fast}}$, and $\Delta t_{\text{slow}} = \Delta t_{\text{fast}}$.

h_{slow} (mm)	Δt_{slow} (s)	Maximum relative difference in APD (%)	Maximum absolute difference in APD (s)
0.5	10^{-4}	0.5053	8.998×10^{-4}
1.0	10^{-4}	2.0930	3.703×10^{-3}
0.1	10^{-3}	0.1396	2.502×10^{-4}
0.5	10^{-3}	0.5873	1.047×10^{-3}
1.0	10^{-3}	1.8750	3.313×10^{-3}

Efficiency of the Algorithm

In Fig. 8 we plot the time required to perform the simulations described in section "Summary of Computations" as a function of h_{slow} and Δt_{slow}. The value of h_{slow} is indicated on the figure: the solid bars represent $\Delta t_{\text{slow}} = 10^{-4}$ s and the open bars represent $\Delta t_{\text{slow}} = 10^{-3}$ s. Also shown is the total time taken to solve the linear systems in each simulation—this value varied by less than 10 s for the different choices of h_{slow} and Δt_{slow} used in these simulations. We see that when $\Delta t_{\text{slow}} = 10^{-4}$ s is used, increasing h_{slow} from 0.1 mm to 0.5 mm reduces computation time by a factor of over 13. Note that the slowest simulation presented here uses the semi-implicit algorithm on a uniform mesh with a uniform timestep described by Whiteley,[28] which is around 8 times more efficient than standard numerical methods for solving the bidomain equations. The algorithm published here therefore gives an increase in computational efficiency of over two orders of magnitude compared to standard numerical methods. Having made this reduction in computation time, roughly one-third of the computation time is now devoted to solving the linear system that arises on each timestep. This linear system is not affected by the algorithm proposed here, and so the time taken for solving these linear systems cannot be reduced. As such the extra efficiency obtained by increasing h_{slow} to 1.0 mm is swamped by the time spent solving the linear system, and so there is no great benefit to using $h_{\text{slow}} = 1.0$ mm rather than $h_{\text{slow}} = 0.5$ mm.

It might be expected that increasing Δt_{slow} from 10^{-4} s to 10^{-3} s would increase computational efficiency by a factor of around 10. We see in Fig. 8 that this is not the case. For the simulations described in this study solving the linear system did not occupy a significant

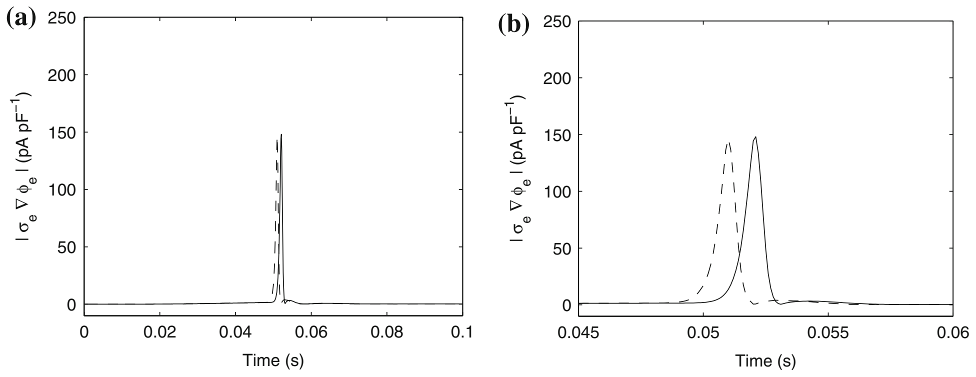

FIGURE 6. The magnitude of the extracellular current at the point ($x = 10$ mm, $y = 10$ mm) calculated when $h_{\text{slow}} = h_{\text{fast}}$, and $\Delta t_{\text{slow}} = 10^{-4}$ s (solid line) and $\Delta t_{\text{slow}} = 10^{-3}$ s (broken line). (a) shows the first 0.1 s of the simulation; (b) shows the rapid variations on an expanded time axis.

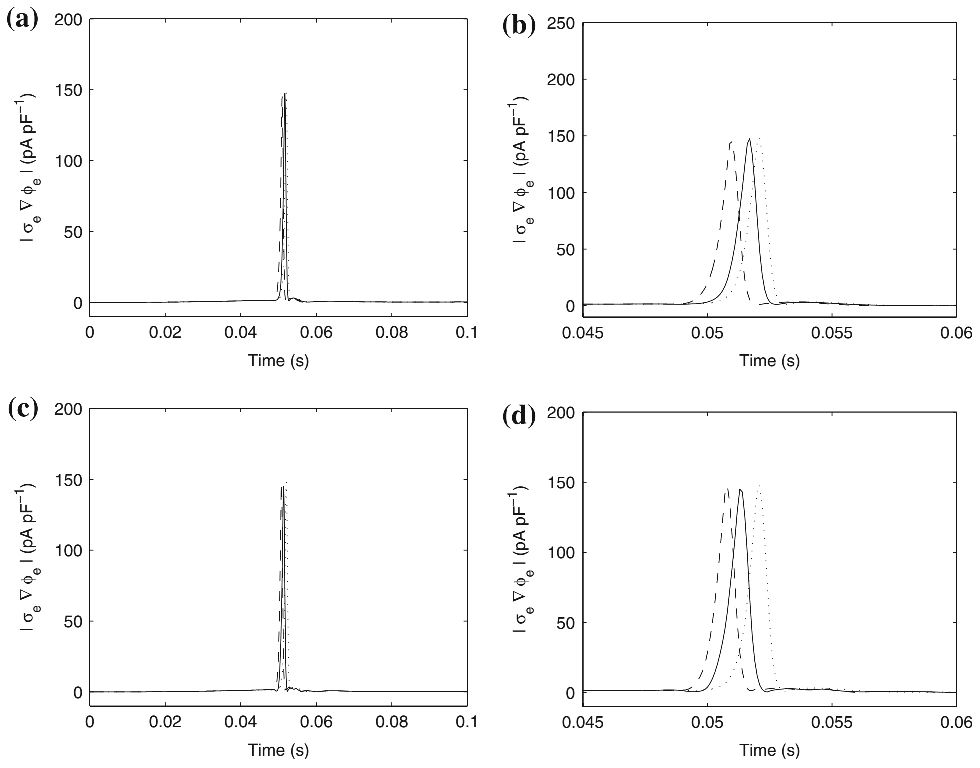

FIGURE 7. The magnitude of the extracellular current at the point (x = 10 mm, y = 10 mm). In all plots the dotted line represents the simulation with $h_{slow} = h_{fast}$, and $\Delta t_{slow} = \Delta t_{fast}$. In (a) the solid line represents the simulation with h_{slow} = 0.5 mm and $\Delta t_{slow} = 10^{-4}$ s, and the broken line represents the simulation with h_{slow} = 0.5 mm and $\Delta t_{slow} = 10^{-3}$; (b) plots the rapid variations seen in (a) on an expanded time axis. In (c) the solid line represents the simulation with h_{slow} = 1.0 mm and $\Delta t_{slow} = 10^{-4}$ s, and the broken line represents the simulation with h_{slow} = 1.0 mm and $\Delta t_{slow} = 10^{-3}$; (d) plots the rapid variations seen in (c) on an expanded time axis.

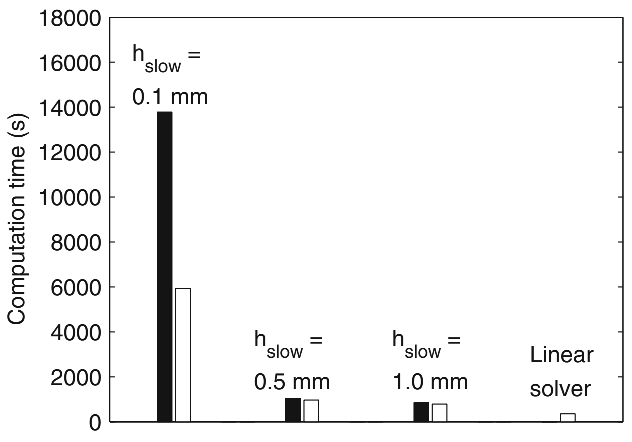

FIGURE 8. Computation time required when using different values of h_{slow} and Δt_{slow}. The value of h_{slow} is indicated on the plot. For each value of h_{slow} the solid bar represents $\Delta t_{slow} = 10^{-4}$ s, and the open bar represents $\Delta t_{slow} = 10^{-3}$ s. Also shown is the total time taken to solve the linear system that is solved on each timestep.

fraction of the computational effort required on each timestep. The main computational effort required on each timestep is: (i) calculating the right-hand-side of the linear system, Eq. (11); and (ii) solving the ordinary differential equations. Changing Δt_{slow} reduces the computational effort required to calculate the right-hand-side of the linear system, allowing a gain in computational efficiency for this part of the computation. We have, however, used a timestep Δt_{fast} to solve the ordinary differential equations as there is no guarantee that a timestep Δt_{slow} is appropriate for all components of this system of ordinary differential equations. This explains the relatively modest increase in computational efficiency obtained when Δt_{slow} is increased from 10^{-4} s to 10^{-3} s. Only when we change h_{slow}—thus requiring the solution of the whole system of ordinary differential equations at fewer points—do we gain an increase in computational efficiency when solving the ordinary differential equations.

We now compare the algorithm presented here with a previously published adaptive algorithm.[29] For the algorithm used in this study we use h_{slow} = 0.5 mm and $\Delta t_{slow} = 10^{-3}$ s. In Fig. 9 we plot the computation time required to simulate 0.35 s, and the computation time required to simulate 1.0 s. Solid bars refer to the algorithm proposed here, open bars to the previously published adaptive algorithm. We see that in both

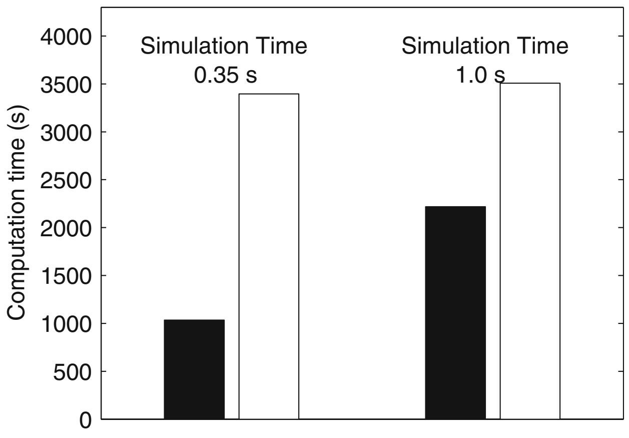

FIGURE 9. Computation time required by the algorithm presented in this study (solid bar) and the adaptive algorithm given by Whiteley[29] (open bar). For the algorithm presented in this study the values h_{slow} = 0.5 mm and $\Delta t_{\text{slow}} = 10^{-3}$ s are used.

cases the algorithm proposed here is more efficient, although the difference is more impressive for the simulation of length 0.35 s where the new algorithm is around three times as fast. The adaptive algorithm performs better on the simulation of length 1.0 s—this is because very little activity happens between 0.35 s and 1.0 s, and the adaptive algorithm performs particularly well under these conditions.

DISCUSSION

We have developed an algorithm that permits an increase in computational efficiency of over two orders of magnitude for the simulations presented in this study. Although we have used the electrophysiological model described by Noble *et al.*[12] in this study, the algorithm may be applied to any model that may be written in the form given by Eqs. (1)–(3). The algorithm presented here is slightly more computationally efficient than a previously published adaptive algorithm for solving the bidomain equations,[29] and has the advantage over this adaptive technique that it does not require that the action potential upstroke is restricted to a small region of space at a given time.

One further advantage of the algorithm presented here is that it is not particularly difficult to implement—the only additional requirement over most currently used software is that two spatial meshes must be generated, together with a method for interpolating between these two meshes. This may easily be achieved by generating the coarser mesh first, and then generating the finer mesh from this coarser mesh. Although this may be tedious when done for the first time, both of these meshes may then be used in as many future simulations as required at no extra cost.

The key to using the algorithm presented here to gain computational efficiency while sacrificing only negligible accuracy is to identify an appropriate partitioning of the total ionic transmembrane current into: (i) those currents that must be calculated on a fine mesh and updated on a short timestep; and (ii) those currents that may be calculated on a coarser mesh and updated on a longer timestep. For the electrophysiological model used in this study we saw in Fig. 1 that the fast sodium current was over an order of magnitude bigger than the sum of the other ionic currents. Specifically, the ionic current with the next largest absolute value is the time-independent potassium current: the maximum magnitude of this current is almost 30 times smaller than the largest magnitude of the fast sodium current. We see from Eqs. (1) and (2) that contributions to I_{ion} with a large magnitude cause rapid changes in V_{m}. This observation allows us to guide an appropriate decomposition of the total ionic transmembrane current: currents with a large absolute magnitude should be calculated on the fine mesh and updated frequently; while those with a small absolute magnitude may be calculated on a coarser mesh and updated less frequently. In particular, a current that does not have a large peak absolute value need not be calculated on the fine mesh and updated frequently even if it does contribute a substantial amount of current over a relatively long time period: the low absolute value of this current will not cause rapid changed in V_m in Eqs. (1) and (2). A sound physiological knowledge of the underlying model will also aid a suitable decomposition. The decomposition may be tested using single cell or one-dimensional simulations in order to verify that calculating certain currents and variables on a coarse mesh does indeed maintain a sufficient level of accuracy.

When partitioning the individual ionic currents it should be noted that some currents included in electrophysiological models do not contribute to the total ionic transmembrane current. For example for the electrophysiological model used in this study[12] one current that has a large magnitude and varies rapidly is calcium release from junctional sarcoplasmic reticulum into other parts of the intracellular space. As this current does not contribute to the transmembrane ionic current it does not appear in Eqs. (1) and (2), and so does not directly affect the spatial propagation of V_{m} and ϕ_{e}. As such, although it has a large magnitude in places, it does not have to be computed on the fine mesh. However, as discussed in section "The Efficient Algorithm", the large magnitude of this current requires that the ordinary differential equations are solved using a timestep that is appropriately small: in this study we solved the ordinary differential equations with a timestep Δt_{fast} to ensure the timestep was suitably small.

The technique described in this study may also be applied to electrophysiological models that contain Markov models of ion channels. These models often have fast transitions between states. Provided the ion channel modeled does not correspond to an ionic current that has a large magnitude there is no need to solve the ordinary differential equations on the fine mesh: the equations may be solved on a coarse mesh, although the timestep used when calculating the numerical solution of these equations should short enough to accurately capture the processes modeled. Should, however, the ion channel modeled correspond to an ionic current that does have a large magnitude, this ionic current should be calculated on the fine mesh.

REFERENCES

[1]Bernus, O., H. Verschelde, and A. V. Panfilov. Modified ionic models of cardiac tissue for efficient large scale computations. *Phys. Med. Biol.* 47:1947–1959, 2002.

[2]Brown, P. N., and H. F. Walker. GMRES on (nearly) singular systems. *SIAM J. Matrix Anal. Appl.* 18:37–51, 1997.

[3]Cherry, E. M., H. S. Greenside, and C. S. Henriquez. Efficient simulation of three-dimensional anisotropic cardiac tissue using an adaptive mesh refinement method. *Chaos* 13:853–865, 2003.

[4]Colli Franzone, P., L. F. Pavarino, and B. Taccardi. Simulating patterns of excitation, repolarization and action potential duration with cardiac bidomain and monodomain models. *Math. Biosci.* 197:35–66, 2005.

[5]Cooper, J., S. W. McKeever, and A. Garny. On the application of partial evaluation to the optimisation of cardiac electrophysiological simulatios. In: Proceedings of ACM SIGPLAN, Charleston, South Carolina, 2006, pp. 12–20.

[6]Hanslien, M., J. Sundnes, and A. Tveito. An unconditionally stable numerical method for the Luo–Rudy I model used in simulations of defibrillation. *Math. Biosci.* 208:375–392, 2007.

[7]Iserles, A. A First Course in the Numerical Analysis of Differential Equations. Cambridge Texts in Applied Maths, Chapter 4, 1996.

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

[9]Keener, J. P., and J. Sneyd. Mathematical Physiology. New York: Springer, Chapter 11, 1998.

[10]Murillo, M., and X.-C. Cai. A fully implicit parallel algorithm for simulating the non-linear electrical activity of the heart. *Numer. Linear Algebr. Appl.* 11:261–277, 2004.

[11]Nickerson, D. P. Modelling Cardiac Electro-mechanics: From CellML to the Whole Heart. PhD Thesis, University of Auckland, New Zealand, 2004.

[12]Noble, D., A. Varghese, P. Kohl, and P. Noble. Improved guinea-pig ventricular cell model incorporating a diadic space, i_{Kr} and i_{Ks}, length- and tension-dependent processes. *Can. J. Cardiol.* 14:123–134, 1998.

[13]Pennacchio, M. The mortar finite element method for the cardiac bidomain model of extracellular potential. *J. Sci. Comput.* 20:191–210, 2004.

[14]Pennacchio, M., and V. Simoncini. Efficient algebraic solution of reaction–diffusion systems for the cardiac excitation process. *J. Comput. Appl. Math.* 145:49–70, 2002.

[15]Plank, G., M. Liebmann, R. Weber dos Santos, E. J. Vigmond, and G. Hasse. Algebraic multigrid preconditioner for the cardiac bidomain model. *IEEE Trans. Biomed. Eng.* 54:585–596, 2007.

[16]Portable Extensible Toolkit for Scientific Computing (PETSc): http://www.mcs.anl.gov/petsc.

[17]Qu, Z., and A. Garfinkel. An advanced algorithm for solving partial differential equation in cardiac conduction. *IEEE Trans. Biomed. Eng.* 46:1166–1168, 1999.

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

[19]Skouibine, K., N. Trayanova, and P. Moore. A numerically efficient model for simulation of defibrilation in an active bidomain sheet of myocardium. *Math. Biosci.* 166:85–100, 2000.

[20]Sundnes, J., G. T. Lines, and A. Tveito. Efficient solution of ordinary differential equations modeling electrical activity in cardiac cells. *Math. Biosci.* 172:55–72, 2001.

[21]Sundnes, J., G. T. Lines, and A. Tveito. An operator splitting method for solving the bidomain equations coupled to a volume conductor model for the torso. *Math. Biosci.* 194:233–248, 2005.

[22]Sundnes, J., B. F. Nielsen, K. A. Mardal, X. Cai, G. T. Lines, and A. Tveito. On the computational complexity of the bidomain and the monodomain models of electrophysiology. *Ann. Biomed. Eng.* 34:1088–1097, 2006.

[23]Tomlinson, K. A., P. J. Hunter, and A. J. Pullan. A finite element method for an eikonal equation model of myocardial excitation wavefront propagation. *SIAM J. Appl. Math.* 63:324–350, 2002.

[24]Trew, M. L., B. H. Smaill, D. P. Bullivant, P. J. Hunter, and A. J. Pullan. A generalized finite difference method for modeling cardiac electrical activation on arbitrary, irregular computational meshes. *Math. Biosci.* 198:169–189, 2005.

[25]Victorri, B., A. Vinet, F. A. Roberge, and J.-P. Drouhard. Numerical integration in the reconstruction of cardiac action potentials using Hodgkin–Huxley-type models. *Comp. Biomed. Res.* 18:10–23, 1985.

[26]Vigmond, E. J., F. Aguel, and N. A. Trayanova. Computational techniques for solving the bidomain equations in three dimensions. *IEEE Trans. Biomed. Eng.* 49:1260–1269, 2002.

[27]Weber dos Santos, R., G. Plank, S. Bauer, and E. J. Vigmond. Parallel multigrid preconditioner for the cardiac bidomain model. *IEEE Trans. Biomed. Eng.* 51:1960–1968, 2004.

[28]Whiteley, J. P. An efficient numerical technique for the solution of the monodomain and bidomain equations. *IEEE Trans. Biomed. Eng.* 53:2139–2147, 2006.

[29]Whiteley, J. P. Physiology driven adaptivity for the numerical solution of the bidomain equations. *Ann. Biomed. Eng.* 35:1510–1520, 2007.