OXFORD CENTRE FOR COLLABORATIVE APPLIED MATHEMATICS

Report Number 10/57

A numerical guide to the solution of the bidomain equations of cardiac electrophysiology

by

P. Pathmanathan, M. O. Bernabeu, R. Bordas, J. Cooper, A. Garny, J. M. Pitt-Francis, J. P. Whiteley, D.J. Gavaghan

Oxford Centre for Collaborative Applied Mathematics
Mathematical Institute
24 - 29 St Giles'
Oxford
OX1 3LB
England

Progress in Biophysics and Molecular Biology 102 (2010) 136–155

Contents lists available at ScienceDirect

Progress in Biophysics and Molecular Biology

journal homepage: www.elsevier.com/locate/pbiomolbio

Review

A numerical guide to the solution of the bidomain equations of cardiac electrophysiology

Pras Pathmanathan [a,*], Miguel O. Bernabeu [a], Rafel Bordas [a], Jonathan Cooper [a], Alan Garny [b], Joe M. Pitt-Francis [a], Jonathan P. Whiteley [a], David J. Gavaghan [a]

[a] Oxford University Computing Laboratory, Wolfson Building, Parks Road, Oxford OX1 3QD, UK
[b] Oxford University Department of Physiology, Anatomy and Genetics, Sherrington Building, Parks Road, Oxford OX1 3PT, UK

ARTICLE INFO

Article history:
Available online 27 May 2010

Keywords:
Cardiac electrophysiology
Bidomain
Computation

ABSTRACT

Simulation of cardiac electrical activity using the bidomain equations can be a massively computationally demanding problem. This study provides a comprehensive guide to numerical bidomain modelling. Each component of bidomain simulations—discretisation, ODE-solution, linear system solution, and parallelisation—is discussed, and previously-used methods are reviewed, new methods are proposed, and issues which cause particular difficulty are highlighted. Particular attention is paid to the choice of stimulus currents, compatibility conditions for the equations, the solution of singular linear systems, and convergence of the numerical scheme.

Contents

* Corresponding author. Tel.: +44 1865 273838; fax: +44 1865 273839.
E-mail addresses: pras@comlab.ox.ac.uk (P. Pathmanathan), miguel.bernabeu@comlab.ox.ac.uk (M.O. Bernabeu), rafb@comlab.ox.ac.uk (R. Bordas), jonc@comlab.ox.ac.uk (J. Cooper), alan.garny@dpag.ox.ac.uk (A. Garny), jmpf@comlab.ox.ac.uk (J.M. Pitt-Francis), jonathan.whiteley@comlab.ox.ac.uk (J.P. Whiteley), gavaghan@comlab.ox.ac.uk (D.J. Gavaghan).

0079-6107/$ – see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.pbiomolbio.2010.05.006

1. Introduction

Tissue-level cardiac electrophysiology is usually modelled using the bidomain equations, a coupled system of equations describing the intracellular and extracellular potential fields through the cardiac tissue. The bidomain equations comprise two partial differential equations (PDEs) coupled at each point in space with a system of ordinary differential equations (ODEs) (see, for example, [Keener and Sneyd, 1998]). The PDEs model electrical potential fields in intra- and extracellular spaces as a reaction-diffusion system. Each system of ODEs represents the concentrations of ions and other variables, such as the proportions of membrane ion channels being open, at the cellular level. The solution of these equations on meshes consisting of up to tens of millions of nodes, and using systems of ODEs at each node in potentially over a hundred variables, is a massively computationally demanding problem. Numerical solution of the bidomain equations requires the efficient and accurate implementation of a number of components, including: the spatial discretisation of the PDEs; the choice of a stable time-discretisation; numerical solution of the ODEs; solution of singular linear systems (and implementation of a suitable preconditioner); as well as careful parallel implementation. In this study, we provide an overview to numerical bidomain modelling by considering techniques for the efficient implementation of each these components, both by reviewing methods that have been used previously, and by proposing new methods and numerically studying their effectiveness. We pay particular attention to issues involved in computational bidomain modelling which are often glossed over in previous papers, namely the choice of stimulus currents, compatibility conditions for the bidomain equations, the solution of singular linear systems, and convergence of the numerical schemes. These issues can cause severe problems in terms of the validity and accuracy of numerical solutions, and in the efficiency of the computations.

The bidomain equations govern the intracellular and extracellular electrical potentials, ϕ_i and ϕ_e, or, equivalently, the extracellular potential ϕ_e and the transmembrane voltage $V = \phi_i - \phi_e$. In the finite element (FE) [Reddy, 1993] spatial discretisation of bidomain equations, we sub-divide the domain into a finite set of N nodes, and triangular or tetrahedral elements. In this paper, we consider only a coupled numerical approach, where V and ϕ_e are computed together, which is usually more efficient than uncoupled methods (see Section 3.1). In this case, let us write the vector of unknowns[1] as $\mathcal{V}^n = [V_1^n, \ldots, V_N^n, \phi_{e1}^n, \ldots, \phi_{eN}^n]^T$, where V_j^n and ϕ_{ej}^n are the voltage and extracellular potential at the j-th node and the n-th timestep. Now, we shall see that one discretisation leads to a linear system of the form

$$A\mathcal{V}^n = \mathbf{b}, \tag{1}$$

at each timestep. We shall see that the matrix A is constant in time and needs only be set up at the beginning of the simulation, but $\mathbf{b}$ depends on $\mathcal{V}^{n-1}$ and the ionic and stimulus currents, and therefore varies with time. There are three stages to computing the solution at each timestep:

1. Integrate the cellular ODEs at each node,
2. Assemble the vector $\mathbf{b}$,
3. Solve the linear system (1).

We will discuss efficient implementation of each of these stages. We begin in Section 2 by stating the governing equations and discussing conditions of compatibility and their implications on the choice of stimulus currents. In Section 3, we consider time and spatial discretisations of the bidomain equations, and discuss efficient assembly of the vector $\mathbf{b}$. Section 4 is concerned with the integration of the ODEs. Next, in Section 5, we study methods of solving the linear system. The matrix A is singular, and we will describe the discrete compatibility conditions and consider methods of solving singular linear systems or of converting the matrix into a non-singular one. Section 6 describes issues relating to parallel implementation. Finally, in Section 7, we study the convergence of our numerical methods in various numerical parameters.

All of the computations in this paper have been run using the cardiac electrophysiology solver in Chaste (**C**ancer, **H**eart **a**nd **S**oft **T**issue **E**nvironment), a general purpose simulation package aimed at multi-scale, computationally demanding problems arising in biology and physiology, developed at the University of Oxford using software engineering techniques imported from the commercial sector (Pitt-Francis et al., 2008, 2009; Bernabeu et al., 2009). The cardiac portion of Chaste is a powerful, efficient, parallel and well-engineered monodomain/bidomain solver. Chaste is open-source and available for download at http://web.comlab.ox.ac.uk/chaste/.

2. The bidomain equations

2.1. The governing equations

Let Ω denote the region occupied by the cardiac tissue. Let us begin with the *parabolic–parabolic* form of the bidomain equations (Keener and Sneyd, 1998)

[1] Note that although we use this 'block' notation for $\mathcal{V}^n$ in this paper, a 'striped' scheme, i.e. $\mathcal{V}^n = [V_1^n, \phi_{e1}^n, \ldots, V_N^n, \phi_{eN}^n]^T$ has been used in practice, for parallelisation reasons. The block notation is used in this paper as it results in a much clearer exposition of the theory and practical implementation.

$$\nabla\cdot(\sigma_{\mathrm{i}}\nabla\phi_{\mathrm{i}}) = \chi\left(\mathcal{C}_{\mathrm{m}}\frac{\partial V}{\partial t} + I_{\mathrm{ion}}(\mathbf{u},V)\right) - I_{\mathrm{i}}^{(\mathrm{vol})}, \tag{2}$$

$$\nabla\cdot(\sigma_{\mathrm{e}}\nabla\phi_{\mathrm{e}}) = -\chi\left(\mathcal{C}_{\mathrm{m}}\frac{\partial V}{\partial t} + I_{\mathrm{ion}}(\mathbf{u},V)\right) - I_{\mathrm{e}}^{(\mathrm{vol})}, \tag{3}$$

$$\frac{\partial \mathbf{u}}{\partial t} = \mathbf{f}(\mathbf{u},V), \tag{4}$$

where σ_{i} is the intracellular conductivity tensor, σ_{e} is the extracellular conductivity tensor, χ is the surface-area-to-volume ratio and $\mathcal{C}_{\mathrm{m}}$ is the membrane capacitance per unit area. $\mathbf{u}$ is a set of cell-level variables, such as ionic concentrations and membrane gating variables, and $I_{\mathrm{ion}} \equiv I_{\mathrm{ion}}(\mathbf{u}, V)$ is the ionic current per unit surface area. Functional forms for I_{ion} and $\mathbf{f}$ are determined by an electrophysiological cell model, examples of which can be found in the CellML Repository.[2] The source terms $I_{\mathrm{i}}^{(\mathrm{vol})}$ and $I_{\mathrm{e}}^{(\mathrm{vol})}$ are the intra- and extra-cellular stimuli per unit volume. Equations (2) and (3) represent local conservation of current in the intra- and extracellular spaces respectively.

The *parabolic-elliptic* form of the bidomain equations is obtained by taking (2), together with the sum of (2) and (3):

$$\chi\left(\mathcal{C}_{\mathrm{m}}\frac{\partial V}{\partial t} + I_{\mathrm{ion}}(\mathbf{u},V)\right) - \nabla\cdot(\sigma_{\mathrm{i}}\nabla(V+\phi_{\mathrm{e}})) = I_{\mathrm{i}}^{(\mathrm{vol})}, \tag{5}$$

$$\nabla\cdot((\sigma_{\mathrm{i}}+\sigma_{\mathrm{e}})\nabla\phi_{\mathrm{e}} + \sigma_{\mathrm{i}}\nabla V) = -I_{\mathrm{total}}^{(\mathrm{vol})}, \tag{6}$$

$$\frac{\partial \mathbf{u}}{\partial t} = \mathbf{f}(\mathbf{u},V), \tag{7}$$

where

$$I_{\mathrm{total}}^{(\mathrm{vol})} = I_{\mathrm{i}}^{(\mathrm{vol})} + I_{\mathrm{e}}^{(\mathrm{vol})}.$$

We stress that it is this sum of the applied stimuli that is in the right-hand side of the elliptic equation (6), not, as commonly stated, just $I_{\mathrm{e}}^{(\mathrm{vol})}$. This affects the physical interpretation of the stimuli (see below), and therefore the physical interpretation of the simulation, and compatibility conditions on the system (see Section 2.2). Setting the right-hand side of (6) to zero (as is common) does not correspond to zero extracellular stimulus. Instead, it corresponds to choosing $I_{\mathrm{e}}^{(\mathrm{vol})} = -I_{\mathrm{i}}^{(\mathrm{vol})}$, i.e. to applying an extracellular stimulus at each point in space where an intracellular stimulus is applied, with a magnitude equal and opposite to the intracellular stimulus. A positive choice of $I_{\mathrm{i}}^{(\mathrm{vol})}$, and therefore negative $I_{\mathrm{e}}^{(\mathrm{vol})}$, corresponds to injecting current into the intracellular space and simultaneously pulling it out of the coincident extracellular space, and thus stimulating the cell whilst enforcing conservation of current. This is a necessary property of the this formulation, but one which is rarely made explicit.

Appropriate boundary conditions for (5) and (6) are the specification of current applied across the boundary

$$\mathbf{n}\cdot(\sigma_{\mathrm{i}}\nabla(V+\phi_{\mathrm{e}})) = I_{\mathrm{i}}^{(\mathrm{surf})}, \tag{8}$$

$$\mathbf{n}\cdot(\sigma_{\mathrm{e}}\nabla\phi_{\mathrm{e}}) = I_{\mathrm{e}}^{(\mathrm{surf})}, \tag{9}$$

where $\mathbf{n}$ is the outward pointing unit normal vector to the tissue, and $I_{\mathrm{i}}^{(\mathrm{surf})}$ and $I_{\mathrm{e}}^{(\mathrm{surf})}$ are the intra- and extra-cellular currents per unit area applied across the boundary. The system of equations (5)–(9) is then closed by specifying suitable initial conditions for V and $\mathbf{u}$ at all points of Ω. Note that we have intentionally stated these equations in their most mathematically general form—with both volume stimuli ($I_{\mathrm{i,e}}^{(\mathrm{vol})}$) and boundary stimuli ($I_{\mathrm{i,e}}^{(\mathrm{surf})}$)—as we will later discuss the most appropriate choice of stimulus current for use in simulations. $I_{\mathrm{i}}^{(\mathrm{vol})}$ and $I_{\mathrm{i}}^{(\mathrm{surf})}$ represent current (per unit volume/surface-area) injected into the intracellular space or passed into the domain through the boundary respectively, and similarly with $I_{\mathrm{e}}^{(\mathrm{vol})}$ and $I_{\mathrm{e}}^{(\mathrm{surf})}$, which are usually equated to applying a shock using electrodes.

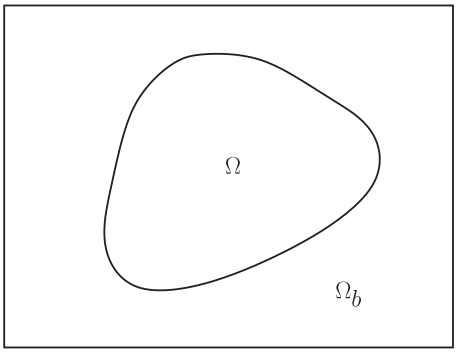

Fig. 1. Domains in a model of cardiac tissue, Ω, contained in a conductive bath, Ω_b.

In the case of cardiac tissue contained in a conductive bath, we introduce a second region Ω_{b}, as shown in Fig. 1. V is defined only in Ω (i.e.only in the tissue) but ϕ_{e} is defined on $\Omega \cup \Omega_{\mathrm{b}}$ (i.e. throughout the tissue and the bath). ϕ_{e} satisfies

$$\nabla\cdot(\sigma_b\nabla\phi_{\mathrm{e}}) = 0 \quad \text{in } \Omega_b, \tag{10}$$

where σ_b is the conductivity of the bath. The boundary conditions are (8) (zero flux of ϕ_{i} across $\partial\Omega$), together with the conditions that the extracellular potential ϕ_{e} and extracellular current[3] $\sigma\nabla\phi_{\mathrm{e}}\cdot\mathbf{n}$ are continuous across the boundary $\partial\Omega$, and (9) replaced by

$$\mathbf{n}\cdot(\sigma_b\nabla\phi_{\mathrm{e}}) = I_{\mathrm{e}}^{(\mathrm{surf})} \quad \text{on } \partial\Omega_b \setminus \partial\Omega. \tag{11}$$

Alternatively, the Dirichlet boundary condition $\phi_{\mathrm{e}} = \phi^*_{\mathrm{e}}$ on some subset of $\partial\Omega_b \backslash \partial\Omega$ could also be specified, corresponding to the application of a grounded electrode.

In this paper, we will mostly restrict ourselves to the basic bidomain equations (5)–(9) when discussing implementation, although these extended equations will be used in the context of electrodes and extracellular stimuli in Section 2.2. Also, we will not explicitly discuss numerical techniques for solving the monodomain equations, a PDE in a single unknown, V, coupled to the cellular electrophysiological ODEs (7), which the bidomain equations reduce to under certain circumstances (see, for example [Keener and Sneyd, 1998]):

$$\chi\left(\mathcal{C}_m\frac{\partial V}{\partial t} + I_{\mathrm{ion}}(\mathbf{u},V)\right) - \nabla\cdot(\sigma\nabla V) = I^{(\mathrm{vol})}, \tag{12}$$

$$\frac{\partial \mathbf{u}}{\partial t} = \mathbf{f}(\mathbf{u},V), \tag{13}$$

where σ is an effective conductivity and $I^{(\mathrm{vol})}$ a stimulus current. However, we will briefly consider which observations and results in

[2] http://models.cellml.org/.

[3] Here σ represents σ_e or σ_b depending on which side of the boundary is being considered.

this paper also apply to the monodomain equation in the Discussion section.

2.2. The singular nature of the problem and the choice of stimuli

2.2.1. Compatibility conditions

In absence of a grounded electrode, the bidomain equations are a naturally singular problem—since ϕ_e only appears in the equations and boundary conditions through its gradient, the solution for ϕ_e is only defined up to a constant. Such problems have compatibility conditions determining whether there are any solutions to the PDEs. This is easily found by integrating (6) over the domain and using the divergence theorem with the boundary conditions (8) and (9):

$$\begin{aligned}\int_\Omega -I_{\text{total}}^{(\text{vol})}(t)\mathrm{d}^3\mathbf{x} &= \int_\Omega \nabla\cdot((\sigma_i+\sigma_e)\nabla\phi_e+\sigma_i\nabla V)\mathrm{d}^3\mathbf{x},\\ &= \int_{\partial\Omega} n\cdot((\sigma_i+\sigma_e)\nabla\phi_e+\sigma_i\nabla V)\mathrm{d}S,\\ &= \int_{\partial\Omega} I_i^{(\text{surf})}(t)+I_e^{(\text{surf})}(t)\mathrm{d}S,\end{aligned}$$

i.e.

$$\int_\Omega I_{\text{total}}^{(\text{vol})}(t)\mathrm{d}^3x+\int_{\partial\Omega} I_i^{(\text{surf})}(t)+I_e^{(\text{surf})}(t)\mathrm{d}S = 0. \tag{14}$$

This equation represents conservation of total current.

Hence, we see that we must choose the four stimuli so that they satisfy this condition. If stimuli are chosen such that this condition is not satisfied, no true solutions of the bidomain equations exist, and any numerical solutions will be spurious.

Correspondingly, the matrix A in the analogous linear system will be singular, and there will be either zero or an infinite number of solutions, and there is an equivalent compatibility condition for the linear system—see (26) in Section 5. Note that when $I_{\text{total}}^{(\text{vol})}$, $I_i^{(\text{surf})}$ and $I_e^{(\text{surf})}$ are specified to be *exactly* zero in floating point arithmetic, this compatibility condition on the linear system will be satisfied exactly without any floating point error (this can be inferred from Sections 3 and 5). In this case, it will be possible to find a solution of the singular linear system accurate to within machine precision. However, if the stimuli are non-zero but such that the constraint is satisfied through cancellation, then errors in numerical integration or floating point errors can lead to a linear system for which the compatibility constraint is not exactly satisfied. In this case, where the compatibility constraint is only approximately satisfied, only an approximate solution can be found. In other words, if there is a small error, $\epsilon>0$, in the compatibility condition, there is some $\delta\equiv\delta(\epsilon)>0$ such that the residual $|A\mathbf{x}-\mathbf{b}|\geq\delta$ for all $\mathbf{x}$, and it will not be possible to solve $A\mathcal{V}=\mathbf{b}$ to machine precision.

2.2.2. Choice of intracellular stimulus

Consider first the case of truly zero extracellular stimuli, i.e. $I_e^{(\text{vol})}=I_e^{(\text{surf})}=0$. Eq. (14) then says that we must have

$$\int_\Omega I_i^{(\text{vol})}(t)\mathrm{d}^3\mathbf{x}+\int_{\partial\Omega} I_i^{(\text{surf})}(t)\mathrm{d}S = 0.$$

It is natural to choose only one of these stimuli to be non-zero, in which case the constraints become $\int_\Omega I_i^{(\text{vol})}\mathrm{d}^3\mathbf{x}=0$ or $\int_{\partial\Omega} I_i^{(\text{surf})}\mathrm{d}S=0$, depending on the choice of stimulus. However, creating a stimulus that satisfies this condition is both physiologically unrealistic (e.g. a stimulus that is positive in one region and negative in another, leading to hyper-polarisation of cells in the negative region), and could be difficult to implement without the linear system compatibility constraint not being satisfied exactly, and the issues of limits on the accuracy of the linear system solution described above.

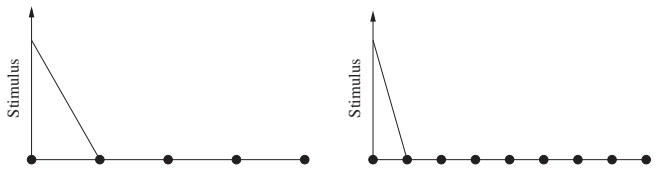

Fig. 2. Stimulus function (as a function of position, at a particular moment in time) on two meshes. If the stimulus is defined to take some positive value on node 0, and be zero at all other nodes, then as the mesh is refined the total stimulus at any time (integral of the stimulus function) tends to zero.

There are two alternatives. One is to ground the system by enforcing a Dirichlet boundary condition on some sub-region of the boundary, so that compatibility is not an issue. The other is specifying a non-zero extracellular stimulus, in which case the obvious choice is (using volume stimuli) choosing $I_e^{(\text{vol})}=-I_i^{(\text{vol})}$, i.e. such that $I_{\text{total}}^{(\text{vol})}=0$. This is what has been implemented, in effect, in any paper which states the bidomain equations with the right-hand side of (6) as zero, with no grounded nodes. Unless stated otherwise, this approach is used in the remainder of the paper.

The question of the best choice of stimulus current is a difficult one, since this bidomain framework applied to the whole heart is a rough approximation to the true physiological system, where the self-activating nature of the system is not modelled and replaced by a regular stimulus current applied at regions where the tissue activation is deemed to begin (e.g. at the end of the Purkinje network). It is unclear how well the use of stimulus currents approximates the true behaviour.

We mention one notable issue associated with a volume intracellular stimulus. Care ought to be taken so that the total applied stimulus converges to some non-zero value as the spatial stepsize of the mesh decreases (for each t). A sufficient condition for this is that the stimulated region has non-zero volume (or non-zero area in two-dimensional simulations, etc). Fig. 2 illustrates how, if the stimulus is only applied to an area of zero volume, the total applied stimulus converges to zero as the stepsize decreases.[4] Note that if the volume over which the stimulus is applied tends to zero, the stimulus magnitude should be increased as stepsize decreases in such a way as to keep $\int I_i^{(\text{vol})}\mathrm{d}\mathbf{x}^3$ constant. This means that in the limit $h\to0$, the stimulus becomes a delta-function—in others words, the volume stimulus has been used as an approximation to a surface stimulus $I_i^{(\text{surf})}$.

2.2.3. Choice of extracellular stimulus

In the case of no intracellular stimuli, $I_i^{(\text{vol})}=I_i^{(\text{surf})}=0$, (14) becomes

$$\int_\Omega I_e^{(\text{vol})}(t)\mathrm{d}^3\mathbf{x}+\int_{\partial\Omega} I_e^{(\text{surf})}(t)\mathrm{d}S = 0.$$

Extracellular stimuli usually correspond to applying a shock using electrodes, in which it is appropriate to consider the bidomain system with a bath, given by (5)–(8), (10) and (11). The

[4] In this figure we assume the stimuli are known nodewise and linear interpolation is used to interpolate the stimuli onto quadrature points in the interiors of elements.

compatibility condition for this case is unchanged, except that the domains of integration become the volume and surface of the whole space:

$$\int_{\Omega\cup\Omega_b} I_e^{(vol)}(t)d^3\mathbf{x} + \int_{\partial\Omega_b\setminus\partial\Omega} I_e^{(surf)}(t)\mathrm{d}S = 0,$$

and only applies if neither electrode is grounded (in which case Dirichlet boundary conditions are used and compatibility is not an issue). Note that working with $I_{total}^{(vol)}$ in the same manner as the previous section is not an option as the extracellular stimulus is usually defined in the bath, for which there is no intracellular space.

Again, it is natural to take one of $I_e^{(vol)}$ or $I_e^{(surf)}$ to be zero, in which case the constraint becomes $\int_{\Omega\cup\Omega_b} I_e^{(vol)}\mathrm{d}^3\mathbf{x} = 0$ or $\int_{\partial\Omega_b\setminus\partial\Omega} I_e^{(surf)}\mathrm{d}S = 0$. Either choice is possible, but it is likely that it will be easier to numerically enforce this constraint with the choice of non-zero $I_e^{(surf)}$.

3. Discretisation and assembly

3.1. Time discretisation and coupled vs uncoupled solvers

The first decision that must be made when developing a numerical scheme for solving the PDE given by (5) is the choice of discretisation in time. A fully explicit discretisation in time is the simplest to code, but the timestep used with this discretisation suffers from a stability restriction, and the choice of timestep used is then dictated by stability constraints, not by the timescales that occur in the problem. On the other hand, a fully implicit discretisation removes any stability restriction on the timestep used, and a timestep that is an order of magnitude greater than would be required by an explicit discretisation can often be used [Whiteley, 2006]. However, noting that the I_{ion} term in (5) is dependent on the solution of the ODEs, (7), we see that a fully implicit discretisation in time would require simultaneously solving the whole system of (5)–(7). This requires solving a massive nonlinear system, with size given by the product of the number of nodes in the computational mesh and the number of dependent variables included in (5)–(7). Such large nonlinear systems are notoriously computationally expensive to solve, and so this method is not a common choice for large scale simulations that model a realistic mammalian cardiac geometry.

The drawbacks to both a fully explicit and fully implicit time discretisation of (5) lead us to choosing a semi–implicit time discretisation, where the diffusion term is treated implicitly but the reaction term explicitly. Suppose we use a constant timestep Δt. Using the notation

$$V^n(\mathbf{x}) = V(\mathbf{x}, n\Delta t), \quad \phi_e^n(\mathbf{x}) = \phi_e(\mathbf{x}, n\Delta t), \quad \mathbf{u}^n(\mathbf{x}) = \mathbf{u}(\mathbf{x}, n\Delta t),$$

the PDEs given by (5) and (6) may be discretised semi-implicitly by

$$\chi\left(\mathcal{C}_m\frac{V^n - V^{n-1}}{\Delta t} + I_{ion}\left(\mathbf{u}^{n-1}, V^{n-1}\right)\right) - \nabla\cdot\left(\sigma_i\nabla\left(V^n + \phi_e^n\right)\right) = I_i^{(vol)}, \tag{15}$$

$$\nabla\cdot\left((\sigma_i + \sigma_e)\nabla\phi_e^n + \sigma_i\nabla V^n\right) = I_{total}^{(vol)}. \tag{16}$$

Equations (15) and (16) may be discretised in space using a variety of numerical techniques, such as the finite difference, element or volume methods. Whatever the choice of spatial discretisation, a linear system of the following form arises (briefly assuming zero stimuli for clarity):

$$\begin{pmatrix} \frac{1}{\Delta t}Q + A_i & A_i \\ A_i & A_i + A_e \end{pmatrix}\begin{pmatrix} \mathbf{V}^n \\ \Phi_e^n \end{pmatrix} = \begin{pmatrix} \frac{1}{\Delta t}Q\mathbf{V}^{n-1} - \frac{1}{\mathcal{C}_m}I_{ion}(\mathbf{u}^{n-1}, \mathbf{V}^{n-1}) \\ 0 \end{pmatrix}, \tag{17}$$

where A_i and A_e are matrices representing the discretisations of $-(\chi\mathcal{C}_m)^{-1}\nabla\cdot(\sigma_i\nabla)$ and $-(\chi\mathcal{C}_m)^{-1}\nabla\cdot(\sigma_e\nabla)$ respectively, Q is the discretisation of the time derivative term, $\mathbf{V}^n$ is the vector of unknown values of V at timestep n, and $\mathbf{\Phi}_e{}^n$ is the vector of unknown values of ϕ_e at timestep n.

Most workers (for example, [Vigmond et al., 2002; Plank et al., 2007; Austin et al., 2006; dos Santos and Dickstein, 2003]) solve (17) by uncoupling this equation and solving the two smaller systems

$$\left(\frac{1}{\Delta t}Q + A_i\right)\mathbf{V}^n = \frac{1}{\Delta t}Q\mathbf{V}^{n-1} - \frac{1}{\mathcal{C}_m}\mathbf{I}_{ion}\left(\mathbf{u}^{n-1}, \mathbf{V}^{n-1}\right) - A_i\Phi_e^{n-1}, \tag{18}$$

$$\left(A_i + A_e\right)\Phi_e^n = -A_i\mathbf{V}^n. \tag{19}$$

Although it seems intuitive that solving the uncoupled system given by (18) and (19) would be more computationally efficient, studies which have compared the efficiency of this approach with solving the coupled system given by (17) have found that solving the coupled system is more efficient [Southern et al., 2009; Sundnes et al., 2005, 2006; Whiteley, 2006]. For two three-dimensional, anatomically realistic, rabbit ventricle simulations [Southern et al., 2009] it was found that solving the coupled system given by (17) is up to 80% faster.

3.2. Spatial discretisation

For clarity, in continuing to describe the numerical method, we now consider a simplified set of equations.

$$\frac{\partial v}{\partial t} - \nabla^2 v - \nabla^2\phi = J(v), \tag{20}$$

$$2\nabla^2\phi + \nabla^2 v = 0, \tag{21}$$

together with zero flux boundary conditions. Here, v is analogous to V, J to the intracellular stimulus together with the ionic current, and (21) is analogous to taking $I_{total}^{(vol)} = 0$, as discussed in Section 2.2.

In order to spatially discretise the PDEs using the finite element method, which is especially well-suited to solving PDEs on irregular geometries such as the heart with derivative boundary conditions, the first step is to convert the equations into their *weak form*, obtained by multiplying the equations by arbitrary test functions and integrating by parts to reduce the system to containing only first-order derivatives. Using test functions ψ, from a suitable (Reddy, 1993) test space W, we have

$$\int_\Omega \frac{\partial v}{\partial t}\psi + \nabla(v+\phi)\cdot\nabla\psi - J(v)\psi\,\mathrm{d}^3\mathbf{x} = 0 \quad \forall\,\psi,$$

$$\int_\Omega 2\nabla\phi\cdot\nabla\psi + \nabla v\cdot\nabla\psi\,\mathrm{d}^3\mathbf{x} = 0 \quad \forall\,\psi.$$

As described in Section 3.1, we then employ a partially implicit time-discretisation, where the linear diffusion terms are treated implicitly but the nonlinear source term explicitly:

$$\int_\Omega \frac{v^n - v^{n-1}}{\Delta t}\psi + \nabla\left(v^n + \phi^n\right)\cdot\nabla\psi - J\left(v^{n-1}\right)\psi\,\mathrm{d}^3\mathbf{x} = 0 \quad \forall\,\psi, \tag{22}$$

$$\int_{\Omega} 2\nabla\phi^n \cdot \nabla\psi + \nabla v^n \cdot \nabla\psi \mathrm{d}^3\mathbf{x} = 0 \quad \forall\ \psi. \tag{23}$$

We then triangulate the domain into a set of N nodes and (triangular/tetrahedral) elements, choose a set of basis functions ψ_1, …, ψ_N (satisfying[5] $\psi_i(\mathbf{x}_j) = \delta_{ij}$, where $\mathbf{x}_j$ is the j-th node) spanning a finite-dimensional subspace of W, write $v^n = \sum v_i^n \psi_i$, $\phi^n = \sum \phi_i^n \psi_i$, and set ψ to be each ψ_i in turn in (22) and (23). Letting $\mathcal{V}^n = [v_1^n, \ldots, v_N^n, \phi_1^n, \ldots, \phi_N^n]^T$, this results in a linear system of size $2N$ of the form

$$A\mathcal{V}^n = \mathbf{b}, \tag{24}$$

as was mentioned in Section 1. Note that the only terms here which contribute to $\mathbf{b}$ are $\int_{\Omega} \frac{v^{n-1}}{\Delta t}\psi \, \mathrm{d}^3\mathbf{x}$ and $\int_{\Omega} -J(v^{n-1})\psi \, \mathrm{d}^3\mathbf{x}$. In the general case with non-zero applied fluxes ($I_{\mathrm{i,e}}^{(\mathrm{surf})}$), there will also be surface integrals of these fluxes contributing to $\mathbf{b}$.

3.3. Matrix-based assembly of $\boldsymbol{b}$

In principle, to assemble A and $\mathbf{b}$ we need to loop over elements in the mesh and compute integrals over each element, using a numerical integration procedure such as Gaussian quadrature, adding the contribution from each element to A and $\mathbf{b}$. A is constant in time (assuming constant Δt), but $\mathbf{b}$ needs to be re-computed (re-assembled) each timestep. Assembling $\mathbf{b}$ in this manner—numerically computing integrals over each element and adding the contribution to $\mathbf{b}$—can make up a significant proportion of the computation time at each timestep, and therefore of the total computation time. However, the time taken to compute $\mathbf{b}$ can be massively reduced by noting that $\mathbf{b}$ is a product of a constant matrix and a vector, as we now describe. Note that the following applies to all reaction-diffusion systems but is especially suitable to bidomain problems.

First, note from (23) that[6] $b_i = 0$ for $N+1 \leq i \leq 2N$, so let us write $\mathbf{b} = [\mathbf{b}^{(1)}, 0, \ldots, 0]^{\mathrm{T}}$, for $\mathbf{b}^{(1)} \in \mathbb{R}^N$. Now, in the particular problem we are solving, J represents the ionic current (and possibly an intracellular volume stimulus current), and assuming the implementation is such that there is a cell at each node (i.e. the ODEs (7) are solved at each node), then J is known *pointwise at the nodes*—because the ionic currents are a function of $\mathbf{u}$ and so only defined at the cells. This should be contrasted with the standard form of a reaction–diffusion equation, where the reaction term is usually given in *functional form*. This apparent complication in bidomain problems is actually very convenient for the assembly of $\mathbf{b}$. In order to integrate functions of J, an interpolation scheme is required, and the obvious choice is $J(\mathbf{x}) = \sum J_i^{n-1}\psi_i(\mathbf{x})$, where $J^{n-1}{}_{\mathrm{i}}$ are the nodal values of the source term at timestep $n-1$. Substituting this into (22) and comparing with (1), we get

$$\mathbf{b}^{(1)} = M\left(\frac{1}{\Delta t}\mathbf{V}^{n-1} + \mathbf{J}^{n-1}\right)$$

where M is the mass matrix ($M_{ij} = \int_{\Omega} \psi_i\psi_j \mathrm{d}^3\mathbf{x}$), $\mathbf{V}^{n-1} = [v^{n-1}{}_1, \ldots, v^{n-1}{}_{\mathrm{N}}]^{\mathrm{T}}$ and $\mathbf{J}^{n-1} = [J_1^{n-1}, \ldots, J_N^{n-1}]^{\mathrm{T}}$. Therefore, to efficiently assemble $\mathbf{b}$, we simply compute M once at the beginning of the simulation, and, at each timestep, set up $\mathbf{J}^{n-1}$ and perform one matrix–vector multiplication. Such matrix-based assembly of $\mathbf{b}$ is also used in Vigmond et al. (2002) and Austin et al. (2006) and others.

Another important advantage of matrix-based assembly of $\mathbf{b}$ is that it removes any need for parallel implementation by the user, assuming a parallel linear system package (such as PETSc, see Sections 5 and 6) is used, since the package will deal with the parallel communication involved in the matrix-vector product, rather than the user having to implement the cross-border communication when assembling on elements containing nodes owned by more than one process. Assembling $\mathbf{b}$ in this manner therefore immediately means that the parallel efficiency of assembling $\mathbf{b}$ will be at least as good as the parallel efficiency of the linear solver.

It should be noted that this method can still be applied in more complex implementations, where cells and nodes do not coincide. One possibility is to have cells (and $J(v)$) defined at the Gaussian quadrature points of the elements—which is equivalent, from a computational point-of-view, to knowing $J(v)$ in functional form, i.e. a general reaction-diffusion problem. Let N_g be the total number of quadrature points. In this case, we have

$$\mathbf{b}^{(1)} = \frac{1}{\Delta t}M\mathbf{V}^{n-1} + P\mathbf{J}^{n-1}$$

where $\mathbf{J}^{n-1}$ is now the source term values at the quadrature points and P is a $N \times N_g$ matrix to be determined which will depend on the precise quadrature scheme used.

Table 1 illustrates the improvements in computational efficiency that are made using this technique. Here, a three-dimensional bidomain simulation on a cuboid domain of 6000 elements is stimulated on one surface and run for 10 ms, with and without matrix-based assembly of $\mathbf{b}$. Without matrix-based assembly (where $\mathbf{b}$ is set-up by looping over each tetrahedral element, using Gaussian quadrature with 8 quadrature points per element to approximate the appropriate integrals over that element, and adding the results to the appropriate entries of $\mathbf{b}$), the time taken to compute $\mathbf{b}$ dominates. When matrix-based assembly, the time taken to compute $\mathbf{b}$ decreases by a factor of 68, with the linear system solve now dominating. The speed-up of the total computation time is 5.4.

4. Solving the ODEs

The solution of the systems of ODEs representing the electrophysiological behaviour of the cells generally accounts for the highest proportion of computational time after solving of the linear system. Two factors contribute to this. Firstly, the systems are stiff, primarily due to the multiple disparate timescales involved in an action potential. Secondly, the systems consist of tens of ODEs, and this number increases as the amount of biological detail included in the models grows. However, the ODE solution is trivially parallelisable, with no spatial interaction except via the PDE solution. Parallel speedup virtually linear in the number of processors is thus achievable for this portion of the simulation.

Table 1
Comparison of the computation time for a 3D bidomain problem with and without matrix-based assembly of $\mathbf{b}$. All times are in seconds.

	ODEs	Setup $\mathbf{b}$	Solve linear system	Total computation time
Without matrix-based assembly	2	41 (82%)	6.1	50
With matrix-based assembly	2 (22%)	0.6	6.2 (67%)	9.2

[5] Here δ_{ij} is the Kronecker delta function, $\delta_{ij} = 1$ if $i = j$ and 0 otherwise.

[6] This assumes zero applied fluxes. If this is not the case then there are surface integrals contributing to $\mathbf{b}$, which can be added on after the procedure described in this section. Computing integrals over the surface of the domain is massively less expensive than computing integrals over the domain.

This section discusses two themes of the efficient solution of the ODEs: the numerical algorithm used (Section 4.1), and additional computational optimisations applied (Section 4.2). Firstly, however, we mention the important question of ensuring that the model simulated is correct. Most of the cell models used for tissue simulations are complex, consisting of many equations and parameters. Coding these from a description in a paper is a time consuming and error prone task, even assuming that the paper does not contain typographical errors or omissions. Thus, model description languages have been developed that provide a computer-readable exchange format for mathematical models. For cardiac models, the most commonly used language is CellML (Lloyd et al., 2004). As mentioned in Section 1, there is a repository of curated CellML models available. Rather than implementing models by hand, it is thus preferable to automatically generate ODE solution code from a CellML model description. Tools such as the CellML API[7] and PyCml[8] (Garny et al., 2008) can assist with this process.

4.1. Numerical approaches

Much research has been done on the numerical solution of ODEs in general (Iserles, 1996). In the context of a single cell simulation, the stiff nature of the system, and the relative lack of activity except at the upstroke, mean that an adaptive implicit solver is the best choice. In the context of a tissue simulation, however, other considerations are also involved.

Work has been done on schemes specific to cardiac electrophysiology. In particular, those equations which represent Hodgkin–Huxley formulations of gating variables can be treated as having an exact exponential solution if the timestep is sufficiently small [Rush and Larsen, 1978].

The remaining ODEs are typically updated using an explicit method such as Runge-Kutta, or even forward Euler, since explicit methods are very cheap compared to implicit methods. The use of an adaptive method is attractive in theory, but difficult to implement efficiently. The additional state required for an adaptive solver results in considerably greater memory requirements for the millions of ODE systems involved in whole-heart simulations. Using different timesteps at different cells also causes load balancing problems for parallel codes.

The primary drawback to explicit methods is that very small timesteps have to be taken throughout the course of the simulation, in order to ensure stability for the upstroke. Some work has been done on the use of implicit methods. Whiteley [Whiteley, 2006; Whiteley, 2007] notes that an efficient backward Euler approach can be obtained by decoupling the ODE systems. If a semi-implicit scheme is used for the PDE solve, v_i^n can be obtained from the solution to the PDE, and many of the ODEs (notably those for Hodgkin–Huxley formulations of gating variables) become linear, and so can be updated directly. Only a small nonlinear system then needs to be solved for each cell. While this does mean each timestep is more expensive than for a forward Euler scheme, the timestep can be an order of magnitude larger since stability is no longer a concern, resulting in an overall speedup (see also [Bernabeu et al., 2009]).

Some investigation has also been done on other adaptive schemes for solving the bidomain equations, which have implications for the solution of the ODEs [Whiteley, 2008; Bernabeu et al., 2009]. These are beyond the scope of this paper, however.

[7] http://www.cellml.org/tools/api/.
[8] https://chaste.comlab.ox.ac.uk/cellml/.

4.2. Computational optimisations

Writing a separate program to simulate each type of cell model, coupled tightly to the simulation environment, and optimising by hand, can produce very efficient code, and this is therefore the route that has traditionally been taken since computation time is at a premium. This has the disadvantages that much work is required for each new cell model, the simulation software becomes very hard to maintain, and confidence in the correctness of the simulations can be low. Since the number and complexity of models in use increases, applying optimisations manually is not a good long term approach, although this may be appropriate in some situations.

Various 'computational' optimisations have also been applied to cardiac cell models. In contrast to the choice of numerical scheme, these techniques transform individual equations within the model into forms that can be calculated more swiftly. The two most prominent techniques are lookup tables (Section 4.2.2) and partial evaluation (Section 4.2.1).

4.2.1. Partial evaluation

Partial evaluation (PE) has long been studied within the computer science community [Jones et al., 1993] as a general approach to automatic optimisation of programs. In its application to cell model simulations [Cooper et al., 2006], expressions are separated into those which need only be computed once, and those which must be recomputed after every timestep of the simulation; this analysis is known as *binding time analysis*. While modern compilers can perform much optimisation along these lines, the complexity of typical ODE solver loops restricts the degree to which this is possible. A partial evaluator produces a new model from the original, in which as much of the model as possible has been precomputed. When simulated, however, the new model will give the same results as the original. PE of CellML models has been implemented in the PyCml tools (Cooper, 2009). Automatic analysis of the model can determine which variables are state or free variables (and hence liable to vary) and which are parameters (assumed to be constant). However, variables may also be annotated as explicitly being dynamic (e.g. for computational steering).

4.2.2. Lookup tables

Lookup tables (LT) are used to pre-compute the values of expressions that would otherwise be repeatedly calculated. Several expressions in most cardiac electrophysiological cell models (notably those controlling the gating variables in Hodgkin–Huxley formulations of ion channels) contain only one variable: the transmembrane potential, V. They also typically contain exponential functions, which are expensive to compute. Under normal physiological conditions, V usually lies between -100 mV and 60 mV, and so a table T can be generated of pre-computed values of each suitable expression for potentials within this range. Then, given any transmembrane potential V within the range, a value t for each expression can quickly be computed using linear interpolation between two entries of the LT:

$$t = T_i + \frac{(T_{i+1} - T_i)(V - V_i)}{V_{i+1} - V_i}, \tag{25}$$

where V_j is the voltage used in computing T_j. (If V lies outside the range, this can either be considered an error condition, or the original equation can be used.) Such interpolation is faster than computing an exponential directly.

The technique has been in use for some time (e.g. [Dexter et al., 1989; Fox et al., 2002]), with the LT code hand-written for each equation; PyCml generates the code automatically, as well as treating subexpressions rather than just whole equations. Note too

that the technique is not restricted to tables indexed on the transmembrane potential, although this is the most common form. Other physiological quantities with known ranges can also potentially be used, such as the intracellular calcium concentration.

Since the use of lookup tables involves an interpolation, this technique introduces an additional source of error. *A posteriori* error analysis may be used to compute a bound on this error, even accumulated over the course of a simulation (paper in preparation). Other workers (e.g. [Fox et al., 2002]) verify the accuracy of a simulation using LT by running the same simulation without LT, perhaps also using a smaller time step. No formal analysis of the resultant numerical error has been given.

The (explicit) mathematics contained within a CellML file is described using MathML, and hence is tree structured. It is thus trivial to construct a recursive algorithm to check any expression (either the whole right hand side of an assignment, or a sub-expression thereof) for suitability for conversion to using an LT. The two key criteria we check are:

1. the only variable used is the transmembrane potential V;
2. the expression contains exponential, logarithmic, or trigonometric functions.

These criteria are checked for each node of the expression tree, starting at the top. The recursion terminates (for a given branch) as soon as both are satisfied, so that maximal subexpressions are matched.

The first criterion for when an LT may be used is very simplistic. Constant variables, and variables whose values depend only on constants, could be included within an expression that is converted to use an LT. The key question is:

is the value known when the LT is generated?

In other words, can the value of the expression be computed (given a value for V) when the LT are generated prior to simulation, or does such a computation require values that are not known until the simulation is in progress? Such an analysis however is basically a binding time analysis. Hence, if PE is applied before the LT analysis then a more complex version of the second criterion is not required. This is because expressions which can be computed prior to simulation are computed by PE and replaced by their value as a constant. The impact of PE on the effectiveness of LT is shown for a range of models in Fig. 3.

The representation of LT within generated code is important for the effectiveness of the technique. In particular, it is crucial to lay out the values in memory such that access to the tables makes good use of the memory cache in typical computer architectures. When performing a simulation, at a given timestep every table will (typically) be accessed, but the same index will be used for each table. Rather than storing each table in its own block of memory, it thus makes sense to arrange the tables together, such that values for a given index are stored contiguously. They are then likely to share a cache line, leading to fewer cache misses and better data throughput. The two approaches are compared in Fig. 3, with 'Base' indicating the layout with all tables combined, and 'Single tables' the use of separate tables for each expression replaced.

When generating C++ code for use with Chaste [Pitt-Francis et al., 2008], PyCml generates two classes: one for the cell model itself, and one following the Singleton design pattern [Gamma et al., 1995] containing the LT for the model. The use of a singleton means that even in a multi-cellular simulation only one instance of the LT will be generated (for a given cell type), thus keeping memory usage to a minimum. The generation of the single LT is performed upon instantiation of a cell model, while (inline) methods are provided to perform linear interpolation on each table. Within the cell model class, calls to these methods replace the appropriate expressions. The latest version of PyCml improves on this approach, since the method calls were found to be adversely impacting performance. 'Row lookups' are now used, in which a single call at the start of each timestep interpolates all the tables and stores the results in a local array. When a value is needed, this array is then indexed. The 'row lookup' columns in Fig. 3 show the effects.

5. Solving the linear system

Next, we turn our attention to solution of the linear system. As mentioned above, the bidomain PDEs only define ϕ_e up to a constant, and correspondingly the matrix A is singular. The compatibility condition for whether a linear system has zero or infinitely-many solutions is: there is no solution to $A\mathcal{V} = \mathbf{b}$ unless $\mathbf{b} \in \ker(A^T)^\perp$ (i.e. $\mathbf{b}\cdot\mathbf{v} = 0 \ \forall \mathbf{v}$ such that $A^T\mathbf{v} = 0$). For the bidomain equations, $\ker(A^T) = \ker(A)$ (by the symmetry of A), and this null space is easy to determine: since ϕ_e is only defined up to a constant, $A\mathbf{v} = 0$ if $\mathbf{v} = [0, \ldots, 0, 1, \ldots, 1]^T$, and the null space is $\ker(A) = \text{span}\{\mathbf{v}\}$. The compatibility condition is therefore that

$$\mathbf{b}\cdot\mathbf{v} = \sum_{i=N+1}^{2N} b_i = 0, \tag{26}$$

which is essentially a discretisation of (14).

We now investigate performance of standard Krylov subspace solvers on this singular problem, as well as considering methods of breaking the singularity. In this section, we will assume the constraint (26) holds exactly. We consider two Krylov solvers, conjugate gradients (CG) [Hestenes and Stiefel, 1952], the standard choice of iterative method for solving large sparse, symmetric, positive-definite problems, and the generalised minimal residual method (GMRES) [Saad and Schultz, 1986], a very common choice of iterative solver for large sparse problems which are non-symmetric. We use the implementations of CG and GMRES in PETSc[9] (a mature library of optimised linear system routines with an efficient parallel implementation, and a very common choice in cardiac modelling). We will consider four approaches: (i) the unaltered Krylov solvers; (ii) the 'null-space' functionality in PETSc, in which components in the null-space are removed from certain vectors computed during CG/GMRES implementation; (iii) specifying the value of ϕ_e at one or several nodes in order to create a non-singular system; and (iv) specifying the value of $\int \phi_e d^3\mathbf{x}$ in order to create a non-singular system. These approaches are described in Sections 5.1–5.4 and compared in Section 5.5. We then compare a selection of solvers and preconditioners in Section 5.6.

5.1. CG and GMRES on singular problems

Note that A is positive-semi-definite (this is easy to show using the block form of A given in (17) and given the positive-semi-definiteness of A_i and A_e (being discretised diffusion operators) and Q, as well as symmetric. The CG method computes search directions $\mathbf{p}_k$, and then uses the quantity $\mathbf{p}_k^T A \mathbf{p}_k$ as the denominator in the computation of two scalars. Breakdown occurs if this quantity is zero. The positive-semi-definiteness of A shows that breakdown cannot occur unless $\mathbf{p}_k$ lies wholly in $\ker(A)$. A stronger statement can be made of GMRES. The results in [Brown and Walker, 1997] imply that, using the symmetry of A, GMRES will not breakdown for any choice of $\mathbf{x}^0$ (the initial guess) and $\mathbf{b}$, and will converge to a least-squares solution. If $\mathbf{b}$ is such that the linear system is

[9] www.mcs.anl.gov/petsc/.

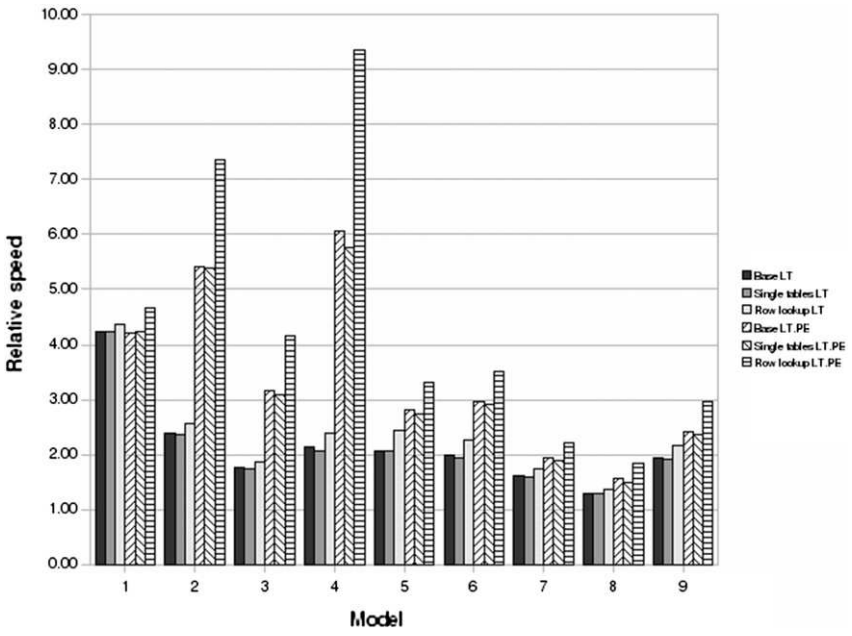

Fig. 3. Single cell simulation timings. Each model was simulated using the forward Euler method with a step size of 0.01 ms (except where a smaller step was required for stability—0.001 ms for [Faber and Rudy, 2000; Fox et al., 2002; ten Tusscher and Panfilov, 2006] and 0.0002 ms for [Bondarenko et al., 2004])), for 1 s of simulated time. These simulations were each run 25 times and the total time recorded. Each bank of simulations was repeated 3 times, and the fastest running time for each was used in calculating speeds relative to the normal model. All simulations used the Intel C++ compiler optimising for Pentium™ 4 processors, and the Intel Math Kernel Library. For all models the LT step size was set to 0.01 mV. The LT columns show speedup due solely to the use of LT, while the LT.PE columns show the combined performance with both optimisations applied. Other aspects are discussed in the text. Models from left to right, ordered by increasing number of ODEs, are: (1) (Noble, 1962); (2) [Luo and Rudy, 1991]; (3) [Fox et al., 2002]; (4) [Garny et al., 2003]. (5) [ten Tusscher and Panfilov, 2006]; (6) [Courtemanche et al., 1998]; (7) [Faber and Rudy, 2000]; (8) [Noble et al., 1998]; (9) [Bondarenko et al., 2004].

compatible, and $\mathbf{x}^0$ is in the range of A (which is satisfied by the choice of $\mathbf{x}^0 = 0$), then GMRES will converge to a solution of $A\mathbf{x} = \mathbf{b}$.

5.2. The PETSc implementations of CG/GMRES using the null-space

PETSc explicitly provides functionality for facilitating the solution of singular linear systems, in which an orthonormal basis for $\ker(A)$ is provided by the user. This is then used to remove the component in the null space of certain vectors computed during the CG or GMRES solves. In the preconditioned CG algorithm, the residual $\mathbf{r}^i = \mathbf{b} - A\mathbf{x}^i$ (where $\mathbf{x}^i$ is the current guess) is computed, and then used to calculate $\mathbf{z}^i = M^{-1}\mathbf{r}^i$, where M^{-1} is the preconditioner. If a basis for the kernel of a singular matrix A is provided, $\mathbf{z}^i$ is then projected onto the orthogonal complement of the kernel (i.e. the component of $\mathbf{z}^i$ in the kernel is removed). For the one-dimensional null-space in bidomain problems, this amounts to computing[10] $\tilde{\mathbf{z}}^i = \mathbf{z}^i - (\mathbf{z}^i \cdot \hat{\mathbf{v}})\hat{\mathbf{v}}$, so that then $\tilde{\mathbf{z}}^i \cdot \mathbf{v} = 0$. In fact, for CG in exact arithmetic, it is straightforward to show that this has no effect on the results of the algorithm: the iterates $\mathbf{x}^i$ with-compared-to-without this modification are identical up to a component in the null-space, and the residuals $\mathbf{r}^i$ (which must always have zero component in the null-space), with-compared-to-without this modification are identical. However, this may not be the case for CG in floating point arithmetic, and the modification prevents accumulation of small components in the null-space growing and affecting the iterates or convergence criterion.

GMRES involves finding, in the i-th iteration, $\mathbf{x}^i \in \mathcal{K}_i$ which minimises $\|\mathbf{b} - A\mathbf{x}^i\|$, where $\mathcal{K}_i = \text{span}\{\mathbf{b}, M^{-1}A\mathbf{b}, (M^{-1}A)^2\mathbf{b}, \ldots, (M^{-1}A)^{i-1}\mathbf{b}\}$ is the i-th Krylov subspace. To do so, an orthonormal basis $\{\mathbf{q}_0, \ldots, \mathbf{q}_{i-1}\}$ of $\mathcal{K}_i$ is computed. The PETSc implementation of GMRES which uses the null-space of A removes any component of $\mathbf{q}_j$ in the kernel (before normalisation), i.e. projects $\mathcal{K}_i$ onto the orthogonal complement of $\ker(A)$.

Both the altered versions of CG and GMRES find, if they converge, the *unique* $\mathbf{x}$ satisfying $A\mathbf{x} = \mathbf{b}$ and $\mathbf{x} \cdot \mathbf{v} = 0$.

5.3. Fixing the value of ϕ_e

A second option is to specify the value of ϕ_e as a Dirichlet boundary condition at one or a group of nodes, which results in a non-singular system, and corresponds to replacing the zero-flux boundary condition with a grounded area through which current can flow. One possibility is to fix the value of ϕ_e at a single boundary node. This is appealing because ϕ_e throughout the domain is defined up to a single constant, and fixing the value of ϕ_e at one point in space fixes this constant. Physically, however, this is not a satisfactory choice as it represents grounding at a single point. More realistic is to fix the value of ϕ_e on a set of nodes on a subsurface $\partial\Omega^{\text{grounded}} \subset \partial\Omega$ of non-zero area, but this models different physics and results in a different solution. Applying a Dirichlet boundary condition can be done by altering the appropriate rows in the linear system (replacing the diagonal values of the matrix with 1 and the off-diagonals with 0). This can be done in a manner which retains the symmetry of A, so that CG can still be used. Note that it is possible to instead apply the boundary condition by using the defined values of ϕ_e explicitly during assembly and reducing the size of the linear system. The convergence properties of the resultant matrices are likely to differ. We only consider the former here.

5.4. Specifying the integral of ϕ_e

A common alternative method of removing the singularity is to choose the arbitrary constant in ϕ_e such that the integral of ϕ_e is 0:

[10] Here $\mathbf{v}$ denotes $\mathbf{v}/\|\mathbf{v}\|$, where $\mathbf{v}$ satisfies $A\mathbf{v} = 0$.

$$\int_{\Omega} \phi_e d^3\mathbf{x} = 0, \tag{27}$$

(or any other fixed value). This equation can be spatially-discretised and used to replace one of the equations in (1) to produce a non-singular linear system.

One problem with this method is that it destroys the sparsity of the linear system. However, this is only in a single row, so it is still possible to allocate only $\mathcal{O}(N)$ bytes of memory for A and perform matrix–vector products in $\mathcal{O}(N)$ operations. A more serious problem is that the alteration breaks the symmetry of A, so that CG is no longer a suitable solver.[11] It is possible however that GMRES on this non-singular, non-symmetric system will perform better than CG on the singular symmetric system.

For simplicity, we replace (27) with the constraint

$$\sum_{j=N+1}^{2N} \mathcal{V}_j = 0,$$

i.e. that the average of the computed ϕ_e values is zero. Note that this is exactly $\mathcal{V}\cdot\mathbf{v} = 0$ (where $A\mathbf{v} = 0$), and therefore the solution is exactly that found by the altered versions of CG/GMRES described in Section 5.2. However, the solution procedure is very different: in one case we incorporate the constraint into the matrix to obtain a non-singular system, in the other we solve the singular (but potentially better conditioned) system and project the solution so that it satisfies the constraint.

5.5. Comparison of the four methods

We compare the different methods by performing a three-dimensional simulation on a slab of tissue of 6000 elements, stimulated on one side and allowed to run for 4 ms. We precondition using an incomplete LU factorisation of A, and solve using an absolute tolerance of 10^{-12}. For the third method we fix ϕ_e at a single corner node (opposite to the side stimulated). Fig. 4(a) displays the average number of iterations, over 400 timesteps, taken by CG and GMRES using each of the four methods, and the convergence history (on the first timestep) of a selection of the methods are plotted in Fig. 4(b). It can be seen that the solvers perform significantly better with the singular system than the non-singular ones. Somewhat unintuitively, the addition of a single Dirichlet boundary condition (fixing ϕ_e) markedly increases the number of iterations required (similar results with a pure-Neumann diffusion problem can be found in [Van der Vorst, 2003], Chap. 10). CG as expected fails in the fourth method, and GMRES here requires more iterations than CG on the singular problem. There is little difference between the unaltered solvers and the ones which used the null space. However, it should be noted that if this experiment is re-run with an absolute tolerance of 10^{-14}, the unaltered GMRES method goes from taking an average of 7.99 iterations to an average of 24.98, whereas the other results only increase slightly. Thus unaltered GMRES can begin to struggle on the singular problem, and it appears that the stabilising effect of removing components from the null space can, in certain cases, make a significant difference. Note that the convergence history of CG with ϕ_e fixed on a whole surface (not just a point) is also given in Fig. 4(b). This performs better than when ϕ_e is fixed at a point, but not enough to prefer it to solving the singular system. Overall the best choice is certainly CG on the singular problem, either with or without the null-space alterations.

[11] An alternative to this is to introduce a Lagrange multiplier λ, and finding the stationary point of , $\frac{1}{2}\mathbf{x}^T A\mathbf{x} - \mathbf{x}^T\mathbf{b} + \lambda\mathbf{v}^T\mathbf{x}$, (where $\mathbf{v}^T\mathbf{x} = 0$ is the discretised constraint), by solving for $\mathbf{y} = (\mathbf{x}, \lambda)$. This will result in a symmetric system of size $2N + 1$. We do not consider this approach in this paper, however.

5.6. A comparison of several linear solver and preconditioners

We conclude this section with a brief, brute-force, comparison of four common linear solvers, with four preconditioners (including the choice of no preconditioner). We use a three-dimensional bidomain simulation on a slab of tissue of 48,000 elements, again stimulated on one side and allowed to run for 4 ms, run in parallel on four processors. The four solvers we consider are: CG; GMRES; the minimal residual method (MINRES) [Paige and Saunders, 1975]; and the SYMMLQ method (Paige and Saunders, 1975). The latter pair of methods are variants of CG that can be applied to symmetric indefinite linear systems. We use the PETSc implementation of each solver, and supply the null-space of the matrix as in Section 5.2. The four preconditioners chosen were the following PETSc preconditioners: no preconditioner; block-Jacobi; successive over-relaxation (SOR); and the additive Schwartz preconditioner. We should note that this is just a small sample of 'black-box' solvers and preconditioners, and also that we have not included the algebraic multigrid (AMG) solver, which can display excellent convergence on bidomain problems [Vigmond et al., 2008].

Table 2 provides the total time spent solving linear systems, for each combination of solver and preconditioner. We see that use of the block Jacobi or additive Schwartz preconditioners reduces the solve time by an order of magnitude, compared to no preconditioner. The SOR preconditioner, however, is not as effective. The block-Jacobi preconditioner is superior to additive Schwartz. With block-Jacobi or additive Schwartz, there is not a big difference between the four solvers, although MINRES is noticeably the least efficient, and CG marginally the best choice of solver. Overall, of these 'black-box' options, CG and block-Jacobi is the best choice on this problem. Note, however, preconditioners designed specifically for the bidomain equations have the potential to be much more effective: in particular, the multi-level additive Schwartz preconditioner proposed in (Pavarino and Scacchi, 2008) is claimed to be scalable and optimal.

6. Parallel implementation and domain decomposition

In this section, we first consider partitioning a mesh of N nodes across p processors, i.e. domain decomposition. This corresponds to the system of linear equations arising from the FE discretisation being partitioned and distributed among the processors available. As our underlying system solver is PETSc, we only consider its partitioning schema. In this package, parallel matrices and vectors are partitioned row-wise, as illustrated below on an example 5-by-5 system

$$\begin{pmatrix} 1 & 2 & 3 & 0 & 0 \\ 4 & 5 & 6 & 7 & 0 \\ 8 & 9 & 10 & 11 & 12 \\ 0 & 13 & 14 & 15 & 16 \\ 0 & 0 & 17 & 18 & 19 \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 6 \\ 22 \\ 50 \\ 58 \\ 54 \end{pmatrix} \tag{28}$$

This illustrates why a striped structure of $\mathcal{V}^n$ (see Section 1) is preferred to a block structure in practice, as the striped scheme allows both unknowns on a given node to be held by the same processor—for example, in a 2-node, 2-processor problem: $(V^n{}_1, \phi^n{}_{e1} \mid V^n{}_2, \phi^n{}_{e2})^T$ compared to $(V^n{}_1, V^n{}_2 \mid \phi^n{}_{e1}, \phi^n{}_{e2})^T$. In the discussion in this section, we assume for clarity that we are only solving a 1-unknown problem (such as the diffusion or monodomain equations), and that there is therefore only a single unknown per node. However, the results in Section 6.2 use the full bidomain problem.

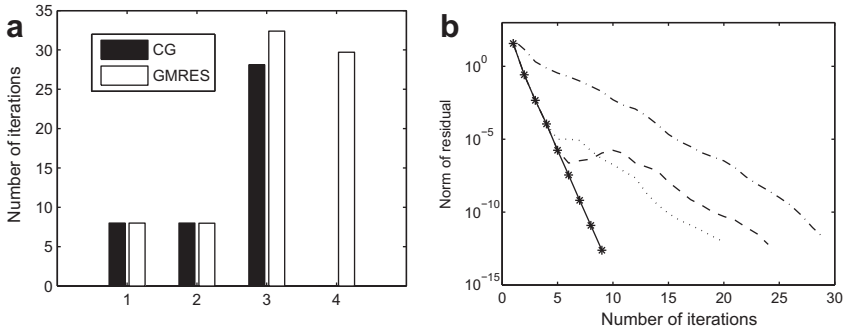

Fig. 4. (a) Average number of iterations per timestep in a 3D bidomain simulation. (1) Unaltered CG and GMRES; (2) PETSc's 'null space' method; (3) Fixing ϕ_e at a point; (4) Specifying $\int \phi_e$ (GMRES only as CG fails in this case). (b) Convergence histories on the first timestep of: unaltered CG (solid line); CG with null-space (stars); CG with ϕ_e at a point (dashed line); GMRES with $\int \phi_e$ specified (dot-dashed line). Also plotted is CG with ϕ_e fixed on a surface (dotted line).

The parallelisation of the different algorithms and data structures involved in a simulation cannot be seen as an isolated problem. It is, indeed, a highly coupled problem. For instance, at the beginning of each timestep the values of V computed in the previous timestep are used to compute the ionic current node-wise (in the ODE solution stage). The vector containing the values of V is a distributed data structure and, therefore, each processor will own part of it. In order to minimise communication, we ensure that a given processor owns the ODE objects corresponding to the entries of V assigned to it. In the example in (28), processor P_0 will own the ODE objects corresponding to nodes $\{0, 1, 2\}$ and processor P_1 will own $\{3, 4\}$. This design ensures no processor will need to fetch non-local data.

When designing the parallelisation schema for the mesh object, one has to look at the dependencies between this data structure and others derived from it. In this case, we have to consider the assembly of the system matrix. In the FE method, the system matrix is assembled element-wise. Each of the elements of the mesh contributes with subblocks that sum to the overall matrix. If an arbitrary element i is made of nodes $\{0, 2, 3\}$ it will contribute with entries in rows $\{0, 2, 3\}$ of the overall system. Thus, it will be advantageous that elements assigned to a processor are primarily made of nodes whose indices correspond to local rows of the matrix. Consider the simple mesh shown in Fig. 5. If the system in (28) is assembled from this mesh, three subblocks will be combined together:

$$A = A_{\mathrm{I}} + A_{\mathrm{II}} + A_{\mathrm{III}}$$

where

$$A_{\mathrm{I}} = \begin{pmatrix} a & b & c & 0 & 0 \\ d & e & f & 0 & 0 \\ g & h & i & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \quad A_{\mathrm{II}} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & j & k & l & 0 \\ 0 & m & n & o & 0 \\ 0 & p & q & r & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \quad A_{\mathrm{III}} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & s & t & u \\ 0 & 0 & v & w & x \\ 0 & 0 & y & z & a' \end{pmatrix}. \tag{29}$$

Table 2
Average time (in seconds), per processor, spent in solving linear systems with four possible solvers and four possible preconditioners, on a test problem run in parallel on four processors.

	CG	GMRES	MINRES	SYMMLQ
No preconditioner	162.5	313.6	208.8	212.9
Block-Jacobi	23.3	24.3	27.7	25.8
SOR	180.8	228.2	214.7	201.5
Additive Schwartz	36.7	38.6	41.0	38.5

All the non-zero entries of A_{I} are owned by P_0, so this processor will own that element. The entries of A_{II} and A_{III} are distributed between P_0 and P_1. In this case, either the subblock is generated in both processors and the non-local entries are ignored or it is generated in one of them and the non-local entries are migrated. In the first approach, some operations are unnecessarily duplicated whereas in the second communication is required. As the number of processors grow the amount of synchronisations required is likely to have a negative impact in the scalability of the partitioning algorithm. We have decided to implement the first approach, since our objective is to minimise communication.

Processor P_0 will assemble subblocks A_{I}, A_{II}, A_{III} and processor P_1 will assemble subblocks A_{II}, A_{III}. In this case, parallel and sequential execution times will be identical, since one of the processors needs to assemble all the elements. This is due to the triviality of the model problem. The same algorithm applied to a full rabbit ventricular mesh of 322, 267 elements returns a distribution of elements for 4 processors presented in Table 3.

The distribution of elements is close to the theoretical optimum of 25%. This is not the case, though, when the same algorithm is run with 16 processors. In that case, each processor owns an average of 8.67% of the elements compared to a theoretical optimum of 6.25%, a 39% overhead. In the following section we will describe how to improve the scalability of the algorithm.

6.1. Graph partitioning and METIS

Let $\mathcal{M} = (\mathcal{N}, \mathcal{E})$ be a computational mesh, where $\mathcal{N}$ is the set of nodes and $\mathcal{E}$ the set of elements. The algorithm presented in the previous section assigns to each processor i a subset of elements $\mathcal{E}_i \subseteq \mathcal{E}$. We define the communication border between two processors j and k as $\mathcal{E}_j \cap \mathcal{E}_k$. Based on this, we can define the following metric for the quality of a partition:

$$Q = \sum_{i \neq j} |\mathcal{E}_i \cap \mathcal{E}_j|$$

We will consider the optimal partition of a mesh the one that minimises this metric while keeping the number of elements

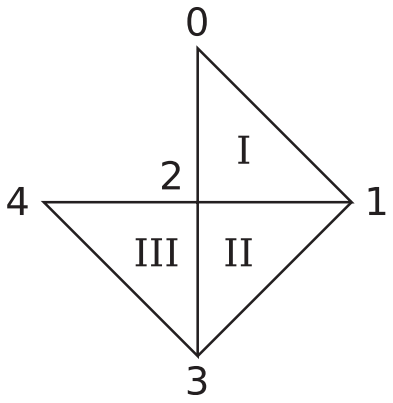

Fig. 5. Simple computational mesh with 3 elements and 5 nodes.

assigned to each processor as close as possible (ideally $|\mathcal{E}_i| = |\mathcal{E}_j|$ $\forall i, j$).

Algorithms that find a good partitioning of highly unstructured graphs are critical for developing efficient solutions for a wide range of problems in many application areas on both serial and parallel computers. The goal of the first condition is to balance the computations among the processors. The goal of the second condition is to minimize the number of elements assembled by multiple processors. Graph partitioning can be used to successfully satisfy these conditions by modelling the FE mesh by a graph and then partitioning it into equal parts.

METIS[12] is a library that implements state-of-the-art algorithms for graph partitioning. It defines a format for representing graphs, an API for handling them, and several partitioning techniques. We can pass to METIS a graph-like representation of our mesh and it will return the partition that minimises Q and balances the number of nodes/elements per processor. For the mesh in Fig. 6(a), the optimal two-processor partition is $P_0 = \{0, 4, 5\}$, $P_1 = \{1, 2, 3\}$. This partition has $Q = 2$ which is the minimum that can be achieved if $|P_0| = |P_1|$ is enforced. Unfortunately, this partition is not compatible with the parallel matrix data structure defined in PETSc, where only consecutive rows can be assigned to a processor. This can be overcome by defining a permutation of the nodes, as shown in Fig. 6(b).

Fig. 7 compares the result of applying the previous partitioning algorithm to the 322, 267-element mesh, with a partition based on the original numbering of the mesh nodes. Results for 16 processors are presented. The average number of nodes per processor in the METIS case is 6.78%, close to the theoretical optimum of 6.25%. Fig. 8, displaying the 4-processor case, shows how nodes assigned to a given processor are rearranged together and communication borders are reduced.

6.2. Performance impact

The use of this new domain decomposition technique has an important impact on the parallel performance of the overall solver. Table 4 displays the speed-up achieved by using METIS in each of the different stages of a bidomain simulation, on a simulation of 10 ms of bidomain activity using a 20 million element mesh with the Luo-Rudy 1991 ionic model (Luo and Rudy, 1991). These simulations were run on a cluster at Fujitsu Laboratories of Europe consisting of 44 compute nodes, each consisting of two dual core Xeon 5160 processors (3.0 GHz) with an InfiniBand interconnect. Fig. 9 plots the speedup of the overall simulation.

[12] http://glaros.dtc.umn.edu/gkhome/views/metis.

Table 3
Parallel distribution of elements for a 322, 267-element ventricular mesh.

Processor ID	Number of local elements	%
0	89,512	27.78
1	101,137	31.38
2	96,636	29.99
3	101,227	31.41

As expected, the new method speeds up the assembly of A and $\mathbf{b}$ by a factor of 91% and 65% for 32 processors. This is due to processors owning less elements and therefore minimising the number of redundant operations. However, a speedup of 80% is also obtained for solving the linear system, for the same number of processors. This improvement was not originally expected and is explained in the following section. Overall, simulations are sped up by 45% to 57% for $p \leq 32$. For $p = 64$, a 77%, 59%, and 67% speedup in assembling A, assembling $\mathbf{b}$ and solving the linear system only translates into 29% global speedup. This is due to scalability problems in stages not covered in this breakdown and requires further investigation.

6.3. Communication reduction in the parallel implementation of CG

The basic computational kernel in the CG algorithm is the matrix-vector product, $\mathbf{y} = A\mathbf{z}$. It is performed at least once in each CG iteration (although depending on the choice of preconditioner more products may be required). This is a highly-coupled operation since each processor does not own enough data to perform it independently. Communication and synchronisation stages are required. Assuming a row-based distribution of the data the operation can be redefined as:

$$\begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{pmatrix} = \begin{pmatrix} A_0 \\ A_1 \\ \vdots \\ A_n \end{pmatrix} \begin{pmatrix} z_0 \\ z_1 \\ \vdots \\ z_n \end{pmatrix},$$

where the sub-matrices A_i are all the rows owned by processor i. Processor i therefore owns A_i and z_i and is in charge of computing subblock y_i:

$$y_i = A_i \begin{pmatrix} z_0 \\ z_1 \\ \vdots \\ z_n \end{pmatrix}$$

It can be easily seen that processor i needs non-local subblocks of $\mathbf{z}$ in order to compute y_i:

$$y_i = (\mathcal{A}_0 \cdots \mathcal{A}_i \cdots \mathcal{A}_n) \begin{pmatrix} z_0 \\ \vdots \\ z_i \\ \vdots \\ z_n \end{pmatrix}$$

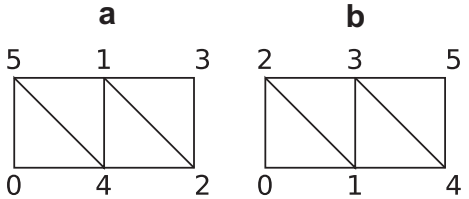

Fig. 6. (a) Simple mesh to be distributed between two processors. (b) Simple mesh with nodes renumbered for use by PETSc.

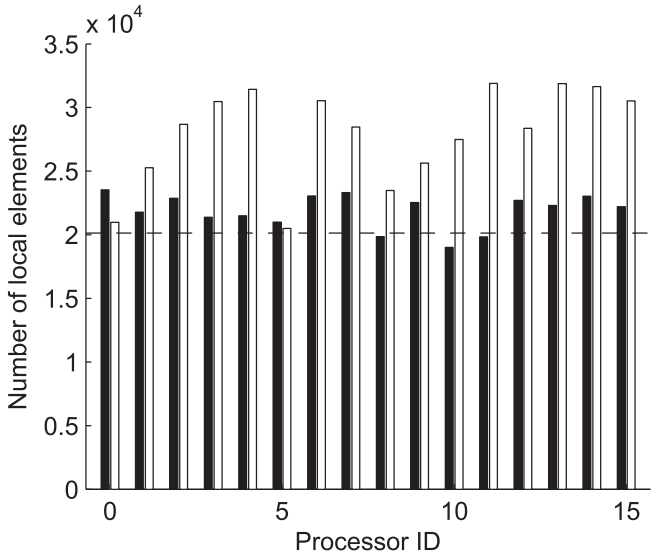

Fig. 7. Comparison between two partition schemes on a 322, 267-element mesh and 16 processors: with METIS (black bars), and without METIS (white bars). The theoretical optimum number of local elements per processor (322, 267/16 = 20, 142) is given as the dashed line.

Subblock $\mathcal{A}_i$ corresponds to the block sitting in the diagonal of A, while $\mathcal{A}_j, i \neq j$ are the off-diagonal subblocks. Communication is required to compute $\mathcal{A}_j z_j$ when $i \neq j$ since z_j is then not local. $\mathcal{A}_i z_i$ is communication-free.

Since $\mathcal{A}_j$ is potentially sparse, not all the entries in z_j, $i \neq j$, need to be sent to processor i. The l-th entry of z_j, $i \neq j$, is required by processor i if and only if there exists k such that $\mathcal{A}_j(k,l) \neq 0$. Minimising the non-zero entries in the off-diagonal subblocks of A minimises the amount of communication required.

Fig. 10 shows the non-zero structure of the matrices assembled for the two two-processor partitions compared throughout this section. It can be seen that minimising the communication border between processors minimises the number of non-zero entries in the off-diagonal blocks of the matrix.

The reason behind this can be better understood if we view the system matrix as the connectivity matrix of the graph associated with the computational mesh. $A_{ij} \neq 0$ represents an edge defined between nodes i and j. Non-zero entries in the diagonal block represent edges defined between local nodes. Non-zero entries in the off-diagonal block represent edges defined across two different processors. Therefore, minimising the communication border between processors minimises the communication required in the matrix–vector product and improves the scalability of the CG algorithm as seen in Table 4. In our model problem, communication is reduced by 33%.

Table 4
METIS vs original partition speedup. Breakdown and totals.

# of processors	Assemble A	ODEs	Assemble **b**	System solution	Total
2	1.20	1	1.09	1.77	1.57
4	1.74	1	1.25	1.77	1.57
8	1.91	1	1.54	1.67	1.50
16	1.81	1	1.59	1.72	1.49
32	1.91	1	1.65	1.80	1.45
64	1.77	1	1.59	1.67	1.29

7. Convergence tests

One area of physiological software development which has been under-documented is that of convergence testing. Most physiological models are nonlinear and span multiple time and space scales. They are therefore generally sensitive to input data, parameter values and numerical methods.

The process of choosing the best space-steps and timesteps involved in producing a realistic simulation and accurate numerical results is something of a 'black art'. Some developers experiment with values until they are able to reproduce a realistically shaped action potential, and then fix on those parameter values. Others may vary a particular parameter (such as space-step) until a run is deemed to be 'repeatable' in the sense that the observed output does not change significantly. In some cases, the process of observing the output may involve plotting two action potentials on the same graph or may involve looking at two three-dimensional 'heat-map' visualisations of the transmembrane potential side-by-side.

In order to design a robust framework for testing convergence, we asked a number of mathematicians, software developers and users of physiological software what they expected to be a suitable metric for measuring the rate of convergence of the solution to the bidomain equations under a variation of a key parameter. Some practitioners took a lenient approach and suggested that

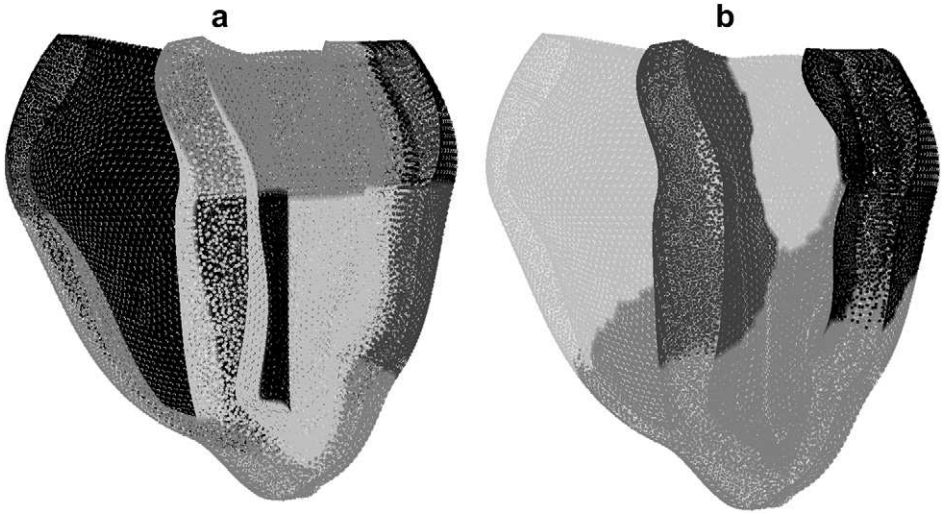

Fig. 8. Graphical representation of the subdomains. Nodes owned by a given processor are displayed in the same colour. (a) Without METIS; (b) with METIS.

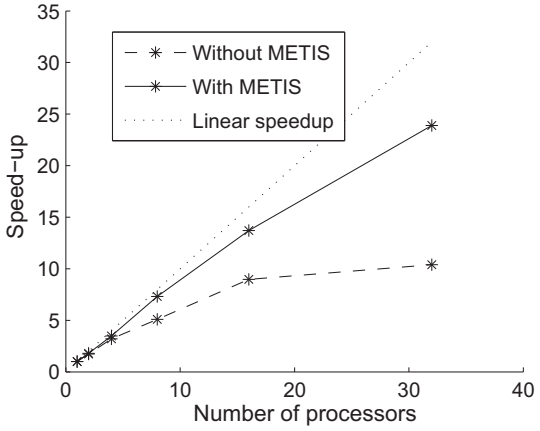

Fig. 9. Parallel perfomance comparison: the speedup with and without METIS against the number of processors, compared to perfect linear speedup.

a physiological output such as the conduction velocity or the action potential duration should be constant to within some precision. Others suggested that they were satisfied if the values of the transmembrane potential were measurably the same at the end of the simulation. (This means of testing convergence was rejected since it is highly influenced by the number of cells which are in their resting phase at the end of the simulation.) Most users of electrophysiological software decided that the most important feature was the shape of the action potential. The experimentalist's observation that the plotted action potential looks the same when graphed can be encoded by the mathematician into a restriction on a 'p-norm' difference of the two solutions. In this study we have used a number of these convergence criteria to show empirically which can be trusted and to inform us of acceptable values of four key convergence parameters: h, the spatial stepsize; Δt_{ODE} , the timestep used in solving the ODEs; Δt_{PDE} , the timestep used in solving the PDEs; and atol, the absolute tolerance used in solving the linear systems.

7.1. Convergence test setup

Our convergence tests use a standardised form of computational experiment. This experiment takes place in a semi-structured mesh of tissue which is in a line (in 1D), square (in 2D) or cube (in 3D) of side length 2 mm. ($0 < x, y, z < 2$ mm). 'Semi-structured' means that the cube (in the 3D case) is divided into smaller cubes in a regular fashion. Each cube in 3D is then divided into 6 tetrahedra of equal size without adding any additional vertices. (In the 2D case, each square is divided into 2 triangles with the direction of the diagonal pointing in a perpendicular direction to its neighbours' diagonals.) The tissue is stimulated on the $x = 0$ mm face of the tissue for the first 0.5 ms of the simulation. The simulation was allowed to run for 8 ms or 400 ms with data at key nodes being stored after every step of the PDE solver. The simulation time of 8 ms allows for a wave to propagate from one edge of the tissue to the other (and thus enough to measure conduction velocity), while the 400 ms allows for all the cells to undergo a complete action potential cycle with a considerable increase in computational expense.

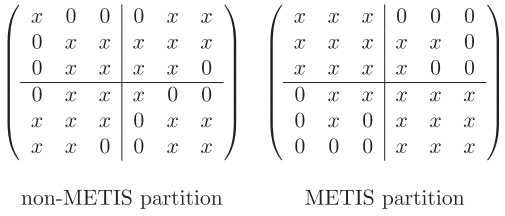

Fig. 10. Non-zero structure comparison.

Although the experiments in this section use an intracellular volume stimulus at nodes on the $x = 0$ plane, our test suites contain convergence experiments with more exotic stimulation protocols (Neumann face stimuli on $x = 0$, for monodomain simulations only, and ramped intracellular current stimuli over the $0 < x < 0.5$ quarter of the mesh). We have found that planar current stimulus protocol to be the most reliable means of commencing wave propagation experiments. However, this choice suffers from the issues described in Section 2.2.2, namely that the total applied stimulus at any moment in time is dependent on the length-scale of the mesh elements and tends to zero as the mesh is refined. However, this issue can be reliably overcome on this regular geometry, as follows. When the mesh is refined by halving the edge lengths of elements then the current stimulus must be doubled. This is because in the first use of the stimulus in the solution of the bidomain equations the current density is interpolated across elements using the finite element scheme; as the volume of the tetrahedra is reduced by a factor of 8, each application of a stimulus is 1/8 as effective. But there are 4 times as many stimulated nodes on the plane $x = 0$. A similar argument applies in 2D (where each stimulus is integrated over an element of quarter the area on each refinement but there are twice as many nodes on the $x = 0$ edge) and in 1D (where the single node stimulus is half as effective on each edge refinement). The upshot of this is that when the typical edge length is reduced from h to h/a then each cell-based stimulus must be increased by the same factor a, regardless of the underlying dimension of the mesh. Empirical demonstration of this rule can be seen in the 1D experiments (Section 7.2, Figs. 11 and 12) and the 2D experiment (Section 7.3, Fig. 13(d)).

The key nodes which we use for measuring physiological output are the first probe point which is one quarter of the way downstream from the stimulus at $x = 0.5$ mm ($y, z = 1$ mm) and a second probe at $x = 1.5$ mm (y, $z = 1$ mm). The space convergence tests start from a base mesh which has an edge-length of $h = 0.5$ mm so that the probe points are coincident with nodes of the mesh at every refinement level.

We also performed these convergence experiments with the alternative stimulation protocol of a ramped intracellular current stimuli over the $0 < x < 0.5$ quarter of the mesh. This protocol mimics the interpolation behaviour of the $x = 0$ plane stimulus on the coarsest mesh. That is, nodes at $x = 0$ are stimulated with a given current value (I_{Stim}) and other nodes in the region are stimulated with the position-dependent current $I_{Stim}(1 - 4x)$. This stimulation protocol is mesh-independent but requires more nodes to be stimulated. It alters the shape of the action potential at the first probe point (since its left neighbours are now stimulated from time $t = 0$). However, the general trends in convergence behaviour under the alternative stimulus protocol are much the same as with the $x = 0$ plane stimulus protocol.

As mentioned above, our convergence tests measure a number of metrics in order to assess convergence. Specifically, the rates of convergence between a simulation run $(k - 1)$ and run k (where some key parameter, for example spatial stepsize, has been refined between the two runs) are

$$\text{rel\_l2\_full(k)} = \sqrt{\frac{\sum_{t\in[0,400\text{ ms}]}\left(v_t^k - v_t^{k-1}\right)^2}{\sum_{t\in[0,400\text{ ms}]}\left(v_t^{k-1}\right)^2}}, \tag{30}$$

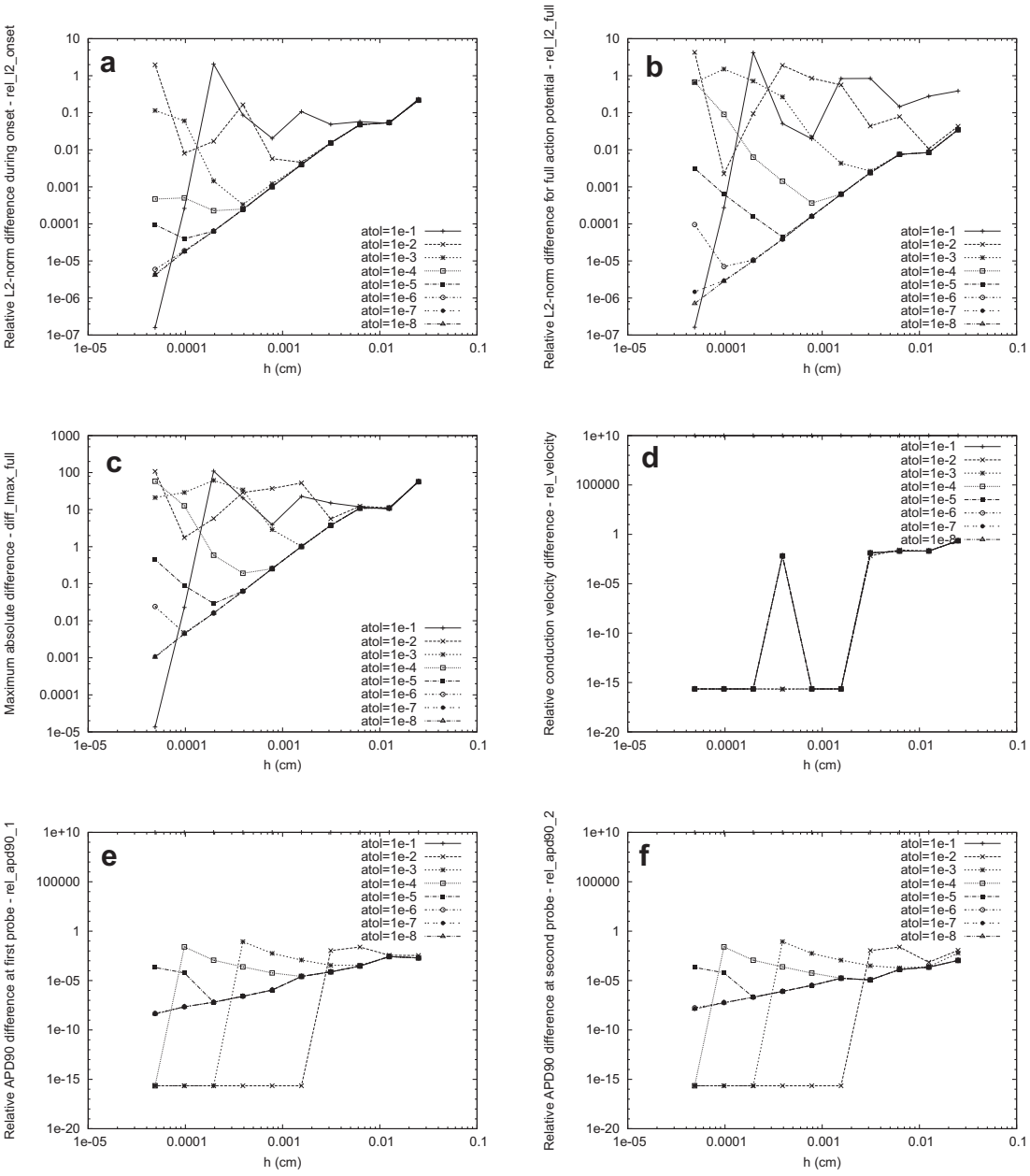

Fig. 11. A set of 1D space-refinement convergence experiments. Each panel shows the variation of a single convergence metric over the full range of tests with each line representing a convergence run with the linear system tolerance set to a given value.

$$\text{rel\_l2\_onset(k)} = \sqrt{\frac{\sum_{t\in[0,8\text{ ms}]}\left(v_t^k - v_t^{k-1}\right)^2}{\sum_{t\in[0,8\text{ ms}]}\left(v_t^{k-1}\right)^2}}, \tag{31}$$

$$\text{diff\_lmax\_full(k)} = \max_{t\in[0,400\text{ ms}]}|v_t^k - v_t^{k-1}|, \tag{32}$$

$$\text{rel\_apd90\_1}(k) = \frac{\text{APD90}_1^k - \text{APD90}_1^{k-1}}{\text{APD90}_1^{k-1}}, \tag{33}$$

$$\text{rel\_apd90\_2}(k) = \frac{\text{APD90}_2^k - \text{APD90}_2^{k-1}}{\text{APD90}_2^{k-1}}, \tag{34}$$

$$\text{rel\_velocity}(k) = \frac{\left|\left(\tau_2^k - \tau_1^k\right) - \left(\tau_2^{k-1} - \tau_1^{k-1}\right)\right|}{\tau_2^{k-1} - \tau_1^{k-1}}; \tag{35}$$

where, in (30)–(32), v_t^k means the transmembrane potential at probe point 2 ($x = 1.5$ mm) at time t on run k. Equations (30) and (31) are relative L_2 norms, whereas (32) is an L_∞ norm. Equations (33) and (34) represent the relative change in the action potential duration at 90% repolarisation (APD90) at probe points 1 and 2, respectively. Equation (35) represents the relative change in a measurement of conduction velocity between the two probe points. The quantity τ_i^k is the time at which the maximum transmembrane upstroke velocity is reached at probe point i on run k. Technically the computation of the conduction velocity also involves division by the distance between the probe points, but this has been abstracted from (35).

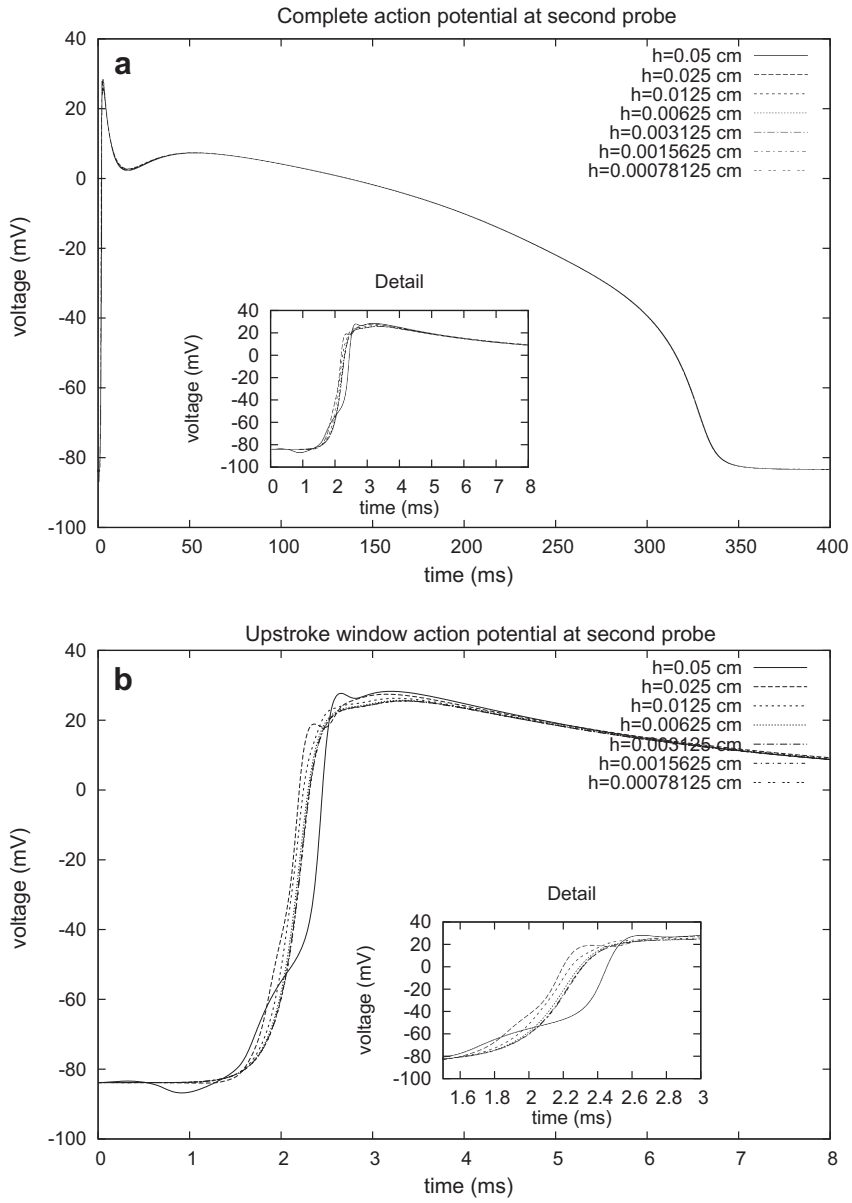

Fig. 12. Plots of the action potential at the second probe point in 1D with the linear system absolute tolerance set to 10^{-4}. (a) The full action potential for a range of mesh refinements with the onset window in the inset panel. (b) The action potential over the onset window for the same range of mesh refinements with the upstroke phase in the inset panel.

Equations (30) and (31) differ only in the window over which the computation is carried out. It was decided that in long running simulations (400 ms) that there should be two measures in the relative L_2 norm, since it is more likely that two computations agree during either the plateau or resting phase of a cell model. If the relative L_2 norm difference quantity is measured both for the whole simulation (rel_l2_full) and for an onset window during which all cells are undergoing excitation (rel_l2_onset), then there is an opportunity of gauging how sensitive this metric is to the time window chosen.

For ease of repeatability, Table 5 shows default physiological parameters used within these Chaste bidomain simulations while Table 6 shows baseline parameters for the two sets of convergence experiments that follow. There is a certain amount of *post hoc* used, since the baseline values for the key parameters are themselves results of successful convergence test: Each key parameter was refined in a convergence experiment until rel_l2_onset $\leq 10^{-2}$ while the other parameters were set to sensible values; then this new value was fed into the other convergence experiments which were re-run for repeatability. Therefore the baseline parameters used in these tests (Table 6) also appear in the table of acceptable parameters (Table 7).

All tests were run on 4 computational cores which reduced the running time of each experiment. In addition to this, our experience shows that at low values of PETSc linear system solver tolerances the solution to the linear algebra problems may vary between sequential and parallel runs of the code. This is presumably because values involved in the iterative solver are subject to machine rounding when sent between processes via MPI messages, but are not rounded when they reside in the memory-space of a single process.

The difference in the choice of cell model solver between the two sets of experiments is merely to demonstrate that if the choice of ODE solver is altered or a new cell model is selected then the

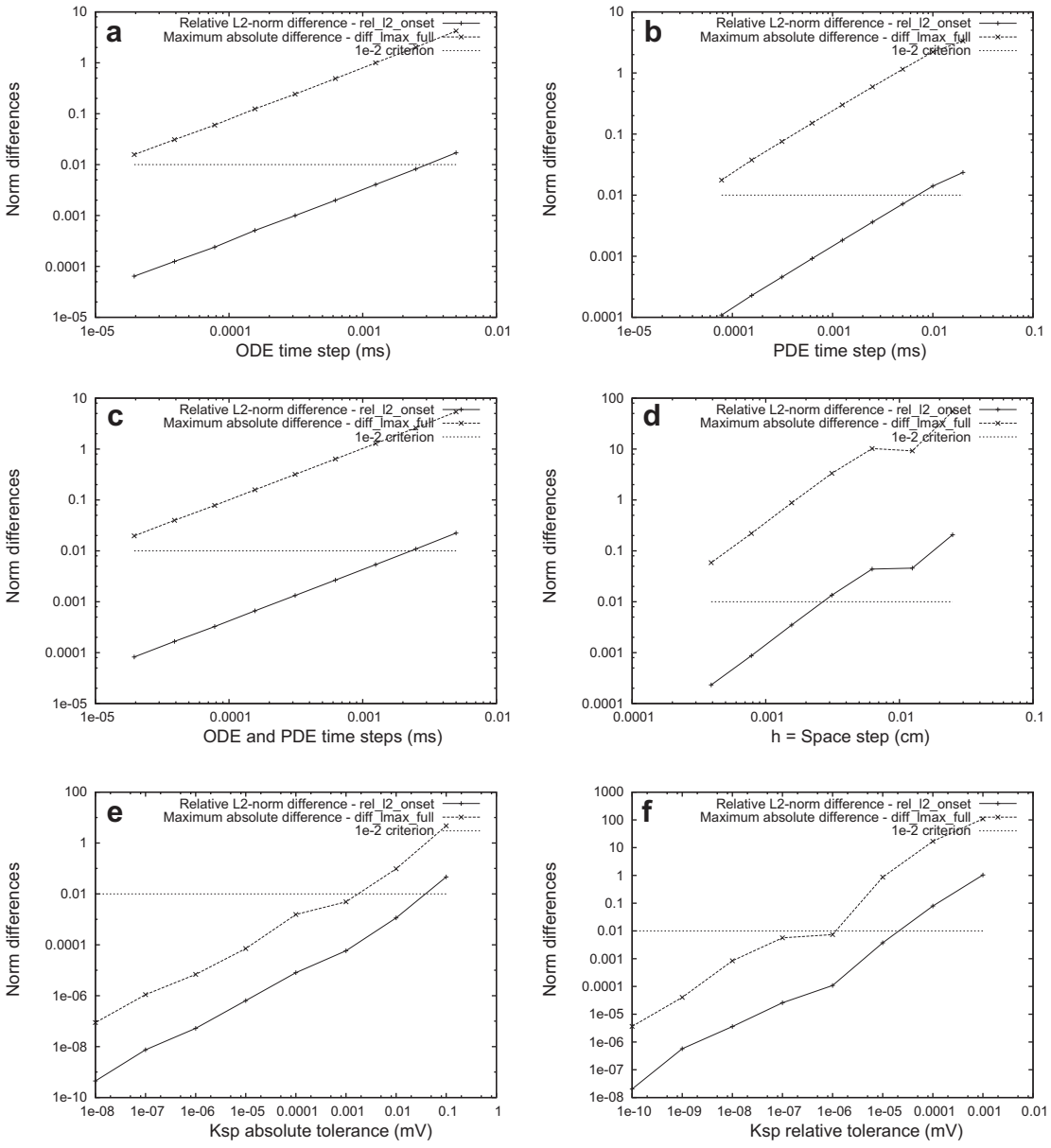

Fig. 13. 2D convergence experiments for a range of key convergence parameters. See Table 6 for details of the parameters which are held constant. Both p-norm metrics available are shown.

convergence for Δt_{ODE} must be re-run in order to verify that the acceptable value of Δt_{ODE} still holds.

7.2. Convergence criteria for 1D spatial refinement experiments

The primary purpose of this set of convergence experiments is to investigate the viability of each of the proposed convergence metrics given in Equations (30)–(35). A 1D mesh refinement experiment was set up in which each of these metrics could be recorded over a full range of mesh refinement from $h = 0.1$ cm down to $h = 7.8 \times 10^{-4}$ cm. In these experiments, we also choose to sweep over a range of tolerances for the linear system solver in order to illuminate the fact that the chosen tolerance is somewhat orthogonal to the other convergence parameters. (This is true in the mesh refinement experiments given here, but is more noticeable in a Δt_{PDE} experiment where halving the timestep will double the number of invocations of the linear system solver during the simulation.) Furthermore, these experiments give us empirical evidence that the intracellular stimulus doubling rule mentioned above is valid in 1D. A similar validation for the stimulus protocol in 2D is contained in Section 7.3.

The value of Δt_{PDE} used for this experiment (10 μs) is slightly bigger than the baseline figure of 5 μs suggested in Table 7. The reason for this selection is that these 1D simulations are dominated

Table 5
Physiological parameters used in all convergence simulations.

Parameter	Value
C_m	0.01 μF mm^{-2}
χ	140 mm^{-1}
σ_i	0.175 mS mm^{-1}
σ_e	0.7 mS mm^{-1}

Table 6
Base-line convergence parameters for convergence simulations.

Parameter	Value in Section 7.2	Value in Section 7.3
Key convergence parameters		
atol	Varied	2×10^{-4}
Δt_{ODE}	2.5 μs	2.5 μs
Δt_{PDE}	10 μs	5 μs
Space step (h)	Varied	1.5625×10^{-3} cm
Other parameters		
Dimension	1D	2D
Mesh size	0.2 cm	0.2 cm × 0.2 cm
Simulation time	400 ms	8 ms
Processes	4 cores	4 cores
Output time step	PDE time step	PDE time step
Cell model	Backward Euler Luo-Rudy 1991	Forward Euler Luo-Rudy 1991
Preconditioner	Jacobi	Block Jacobi
Linear solver	CG	CG

by output and a coarsening in Δt_{PDE} drastically reduces the running times.

Fig. 11 shows the results of these mesh-refinement experiments with atol being swept across a wide range of reasonable values. It is evident from these log-log plots that when the linear system tolerance is too loose (atol $\simeq 0.1$) then convergence cannot reliably be achieved in any metric. As the tolerance is reduced down to $atol = 10^{-8}$ each of the p-norm type metrics has a straight line segment, with the length of the straight part increasing as the tolerance is decreased. The gradient of this line can be measured to be about 1.95 on these log-log plots, which is as expected—convergence for a numerical approximation to the diffusion equation is quadratic, $O(h^2)$. Fig. 11(a) and (b) illustrates clearly the need to reduce the absolute tolerance (as defined by PETSc) when refining the mesh.

While the p-norm metrics all exhibit quadratic convergence, the APD and conduction velocity metrics are less useful. The metric rel_velocity is perhaps the least illuminating. It shows a small period of convergence on the coarsest meshes but then converges to a consistent figure because the values of τ_i^k in (35) can only be measured or extrapolated from the nearest PDE timestep. After this level of convergence has been achieved the value of rel_velocity is effectively zero, although it has been rounded up to the machine precision for the log-log plots. Similar behaviour can be seen in the plots for the APD metrics where it appears that at all but the lowest linear system tolerance levels, the APD90 solutions diverge slightly from the straight line behaviour before converging to within machine precision.

7.2.1. Choice of onset window

Since most metrics are measured from the action potential at the second probe point it is helpful to investigate the convergence behaviour at that node in more detail. Fig. 12 shows graphs of the action potential at that node in the 1D experiments when $atol = 10^{-4}$ for a range of mesh step sizes. It is evident from the first graph over the full simulation that the experimenter's requirement that the graphs should 'look the same' is trivially met and that a plot at this scale gives no indication as to whether the experiment is converging. The corresponding plot of rel_l2_full (in Fig. 11(b)) shows that there is indeed convergence in the difference between the graphs, but it cannot be viewed at this scale.

Fig. 12(b) shows the detail during the onset window when all the cells in the simulation are undergoing excitation. Here, the plots for the 4 coarsest meshes are individually visible, while the plots from the 3 finest are coincident. The values of metrics for the difference between the plot at $h = 0.003125$ cm and $h = 0.0015625$ cm are rel_l2_onset $\simeq 4 \times 10^{-3}$, rel_l2_full $\simeq 7 \times 10^{-4}$ and diff_lmax_full $\simeq 1$ mV.

From these data rel_l2_onset $\simeq 10^{-2}$ was chosen as a suitable basic convergence criterion. This can be interpreted as 'a 1% difference in graphs' during excitation. Its use is subjective, since the convergence plots in Fig. 11 show that there is a range of metrics over which quadratic convergence in space-step may be maintained. However, if another metric is required by a user then there is enough evidence to be able to predict a required mesh size to meet convergence.

Table 7
Approximate acceptable parameters for a given accuracy.

Parameter	rel_l2_ onset $\leq 10^{-2}$	rel_l2_ onset $\leq 5 \times 10^{-3}$	rel_l2_ onset $\leq 10^{-3}$
ODE timestep	2.5 μs	1.25 μs	0.3125 μs
PDE timestep	5.0 μs	2.5 μs	0.625 μs
Space-step	1.5625×10^{-3} cm	1.5625×10^{-3} cm	7.8125×10^{-4} cm

7.3. Convergence experiments for various parameters

We can now perform a set of convergence experiments for a range of key convergence parameters in 2D. For each convergence experiment the plots show rel_l2_onset and diff_lmax. rel_l2_full is identical to rel_l2_onset since the these simulations were run for 8 ms. If we were to run these simulations for a full action potential, then the 1D experiments in Section 7.2 predict that diff_lmax, rel_l2_onset and rel_l2_full are well correlated.

Fig. 13 shows the outputs from these experiments. It is noticeable that diff_lmax and rel_l2_onset are well correlated. The log-log gradient of the straight line section of Fig. 13(d) is about 2, demonstrating in 2D that there is $O(h^2)$ spatial convergence. The log-log gradient of each of the other graphs is close to 1 (1 ± 0.01), demonstrating that there is linear dependence on PDE timestep, ODE timestep and linear system tolerance throughout these experiments.

When the ODE and PDE experiments are combined (Fig. 13(c)) in a convergence experiment which has one ODE step per PDE step, we find that the convergence is dominated by Δt_{ODE}. This is expected, since the required Δt_{ODE} needed to reach the convergence criterion is smaller than Δt_{PDE} (see Table 7). The graph for the combined experiment (Fig. 13(c)) compares well to that for the Δt_{ODE} experiment (Fig. 13(a)) but with each abscissa value slightly higher. This is because of the cumulative effects of the PDE and ODE errors in this experiment.

The linear system tolerance convergence experiments show that there is a linear dependence on the tolerance. The relative tolerance graph (Fig. 13(f)) is not as straight as the atol graph (Fig. 13(e)) because of the variation in the sizes of the values of ϕ_e and V_m during each simulation. At a time when all cells are at rest ($V_m \simeq -90$ mV), the linear system solution to a given relative tolerance may be looser than convergence to the same relative tolerance when the nodes are in the plateau phase ($V_m \simeq 0$ mV). Also, since ϕ_e is only determined up to a constant one must be careful to make sure that its solution can be bounded to within a similar range to that of V_m, if one is to use a relative tolerance.

Table 7 shows the range of key timesteps and space-steps used to produce a convergence criterion of *rel_l2_onset* $\leq 10^{-2}$ (as used for our base-line parameters). Two stricter convergence criteria are also given for comparison.

8. Discussion

Simulation of cardiac electrophysiological activity on the tissue or organ level with the bidomain equations is, despite the relatively

simple form of the equations (a pair of parabolic PDEs, or a parabolic PDE and an elliptic one), linear except for the reaction terms, coupled to a set of ODEs), an extremely computationally-demanding procedure, largely due to the disparate spatial and temporal scales involved, the singular nature of the PDEs, and the complexity of the physiological cell models that have been proposed. In this paper, we have investigated several of the stages involved in such computational models. We began by describing conditions of compatibility for the bidomain equations, and discussed their implications on possible choices of stimulus current. We have described efficient and stable spatial and temporal discretisations, stated a method for fast assembly of the load vector **b**, and described techniques for efficiently solving (a very large number of) ODEs. We investigated methods for solving the singular linear system, discovering that the conjugate gradient method on the original singular matrix is the most effective (compared to methods of converting the matrix into a non-singular matrix), and that the null-space adaptation in linear solvers in the library PETSc can have a useful stabilising effect. We also discussed effective parallel implementation, as well as devising reliable convergence experiments and determining suitable choices of each of the main numerical parameters (stating quantitatively the expected error in given norms with these choices).

Several, but not all, of these results carry through to simulation of electrophysiologial activity using the monodomain equations, (12) and (13). Since the PDE in the monodomain equations is a single reaction-diffusion equation in a single unknown, V, the equation is not singular, so there are no compatibility conditions on the stimulus current, and any volume or surface stimulus is possible, as long as it does not hyper-polarise or over-polarise any cells. This also means that the linear system is non-singular, so there is no need to consider different solvers—since the system is also symmetric, conjugate gradients is the obvious choice. However, the semi-implicit time-discretisation applies equally to the monodomain equations as the bidomain equations, and the matrix-based assembly of the load vector **b** is just as important for the monodomain equations. The discussions of methods of solving the ODEs and parallel implementation also fully apply to monodomain problems. Although not verified, it is expected that the choice of numerical parameters proposed in the convergence experiments for the bidomain equations are also suitable choices for monodomain equations.

Acknowledgements

PP is supported by the EPSRC-funded OXMOS project *New frontiers in the mathematics of solids*, (grant reference EP/D048400/1). JC is supported by the European Community's Seventh Framework Programme [FP7/2007-2013] under grant agreement 223920 (VPH NoE). AG is funded through the preDiCT and euHeart projects (numbers 224381 and 224495, respectively) which are both supported by the European Commission, DG Information Society, through the Seventh Framework Programme of Information and Communication Technologies. JPW is supported by Award No. KUK-C1-013-04, made by King Abdullah University of Science and Technology (KAUST).

References

Austin, T., Trew, M., Pullan, A., 2006. Solving the cardiac bidomain equations for discontinuous conductivities. IEEE Trans. Biomed. Eng. 53 (7), 1265–1272.

Bernabeu, M., Bordas, R., Pathmanathan, P., Pitt-Francis, J., Cooper, J., Garny, A., Gavaghan, D., Rodriguez, B., Southern, J., Whiteley, J., 2009. Chaste: incorporating a novel multi-scale spatial and temporal algorithm into a large-scale open source library. Phil. Trans. Roy. Soc. A 367 (1895), 1907–1930.

Bondarenko, V., Szigeti, G., Bett, G., Kim, S.-J., Rasmusson, R., 2004. A computer model of the action potential of the mouse ventricular myocytes. Am. J. Physiol. Heart Circ. Physiol. 287, H1378–H1403.

Brown, P., Walker, H., 1997. GMRES on (nearly) singular systems. SIAM J. Matrix Analysis Appl. 18 (1), 37–51.

Cooper, J., McKeever, S., Garny, A., 2006. On the application of partial evaluation to the optimisation of cardiac electrophysiological simulations. In: PEPM '06: Proceedings of the 2006 ACM SIGPLAN Symposium on Partial Evaluation and Semantics-based Program Manipulation. ACM Press, New York, NY, USA, pp. 12–20.

Cooper, J., 2009. Automatic validation and optimisation of biological models. Ph.D. thesis, University of Oxford.

Courtemanche, M., Ramirez, R., Nattel, S., 1998. Ionic mechanisms underlying human atrial action potential properties: insights from a mathematical model. Am. J. Physiol. Heart Circ. Physiol. 275 (1), H301–H321.

Dexter, F., Saidel, G., Levy, M., Rudy, Y., 1989. Mathematical model of dependence of heart rate on tissue concentration of acetylcholine. Am. J. Physiol. Heart Circ. Physiol. 256 (2), H520–H526.

dos Santos, R., Dickstein, F., 2003. On the influence of a volume conductor on the orientation of currents in a thin cardiac tissue. Lecture Notes Comput. Sci., 111–121.

Faber, G., Rudy, Y., 2000. Action potential and contractility changes in $[Na^+]_i$ overloaded cardiac myocytes: a simulation study. Biophys. J. 78 (5), 2392–2404.

Fox, J., McHarg, J., Gilmour, R., 2002. Ionic mechanism of electrical alternans. Am. J. Physiol. Heart Circ. Physiol. 282, H516–H530.

Gamma, E., Helm, R., Johnson, R., Vlissides, J., 1995. Design Patterns: Elements of Reusable Object-Oriented Software. Addison–Wesley.

Garny, A., Kohl, P., Hunter, P., Boyett, M., Noble, D., 2003. One-dimensional rabbit sinoatrial node models: benefits and limitations. J. Cardiovasc. Electrophysiol. 14 (s10), S121–S132.

Garny, A., Nickerson, D., Cooper, J., dos Santos, R., McKeever, S., Nielsen, P., Hunter, P., 2008. CellML and associated tools and techniques. Phil. Trans. Roy. Soc. A 366 (1878), 3017–3043.

Hestenes, M., Stiefel, E., 1952. Methods of conjugate gradients for solving linear systems. J. Res. Natl. Bureau Standards 49 (6), 409–436.

Iserles, A., 1996. A First Course in the Numerical Analysis of Differential Equations. Cambridge University Press.

Jones, N., Gomard, C., Sestoft, P., 1993. Partial Evaluation and Automatic Program Generation. Prentice Hall.

Keener, J., Sneyd, J., 1998. Mathematical physiology. In: Interdisciplinary Applied Mathematics, vol. 8. Springer.

Lloyd, C., Halstead, M., Nielsen, P., 2004. CellML: its future, present and past. Prog. Biophys. Mol. Biol. 85, 433–450.

Luo, C., Rudy, Y., 1991. A model of the ventricular cardiac action potential: depolarization, repolarization, and their interaction. Circ. Res. 68, 1501–1526.

Noble, D., 1962. A modification of the Hodgkin–Huxley equations applicable to Purkinje fibre action and pacemaker potentials. J. Physiol. 160, 317–352.

Noble, D., Varghese, A., Kohl, P., Noble, P., 1998. Improved guinea-pig ventricular cell model incorporating a diadic space, i_{Kr} and i_{Ks}, length- and tension-dependent processes. Can. J. Cardiol. 14 (1), 123–134.

Paige, C., Saunders, M., 1975. Solution of sparse indefinite systems of linear equations. SIAM J. Numer. Anal., 617–629.

Pavarino, L., Scacchi, S., 2008. Multilevel additive Schwarz preconditioners for the Bidomain reaction-diffusion system. SIAM J. Sci. Comput. 31, 420.

Pitt-Francis, J., Bernabeu, M., Cooper, J., Garny, A., Momtahan, L., Osborne, J., Pathmanathan, P., Rodriguez, B., Whiteley, J., Gavaghan, D., 2008. Chaste: using agile programming techniques to develop computational biology software. Phil. Trans. Roy. Soc. A 366 (1878), 3111–3136.

Pitt-Francis, J., Pathmanathan, P., Bernabeu, M., Bordas, R., Cooper, J., Fletcher, A., Mirams, G., Murray, P., Osborne, J., Walter, A., Chapman, S., Garny, A., van Leeuwen, I., Maini, K., Rodriguez, B., Whiteley, J., Byrne, H., Gavaghan, D., 2009. Chaste: a test-driven approach to software development for biological modelling. Comput. Phys. Commun. 180 (12), 2452–2471.

Plank, G., Liebmann, M., dos Santos, R., Vigmond, E., Haase, G., 2007. Algebraic multigrid preconditioner for the cardiac bidomain model. IEEE Trans. Biomed. Eng. 54 (4), 585–596.

Reddy, J., 1993. An Introduction to the Finite Element Method. McGraw–Hill.

Rush, S., Larsen, H., 1978. A practical algorithm for solving dynamic membrane equations. IEEE Trans. Biomed. Eng. BME 25 (4), 389–392.

Saad, Y., Schultz, M.H., 1986. GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems. SIAM J. Sci. Comput. 7, 856–869.

Southern, J., Plank, G., Vigmond, E., Whiteley, J., 2009. Solving the coupled system improves computational efficiency of the bidomain equations. IEEE Trans. Biomed. Eng. 56 (10), 2404–2412.

Sundnes, J., Lines, G., Tveito, A., 2005. An operator splitting method for solving the bidomain equations coupled to a volume conductor model for the torso. Math. Biosci. 194 (2), 233–248.

Sundnes, J., Nielsen, B., Mardal, K., Cai, X., Lines, G., Tveito, A., 2006. On the computational complexity of the bidomain and the monodomain models of electrophysiology. Ann. Biomed. Eng. 34 (7), 1088–1097.

ten Tusscher, K., Panfilov, A., 2006. Alternans and spiral breakup in a human ventricular tissue model. Am. J. Physiol. Heart Circ. Physiol. 291 (3), H1088–H1100.

Van der Vorst, H., 2003. Iterative Krylov Methods for Large Linear Systems. Cambridge Univ Pr.

Vigmond, E., Aguel, F., Trayanova, N., 2002. Computational techniques for solving the bidomain equations in three dimensions. IEEE Trans. Biomed. Eng. 49 (11), 1260–1269.
Vigmond, E., dos Santos, R., Prassl, A., Deo, M., Plank, G., 2008. Solvers for the cardiac bidomain equations. Prog. Biophys. Mol. Biol. 96 (1–3), 3–18.
Whiteley, J., 2006. An efficient numerical technique for the solution of the monodomain and bidomain equations. IEEE Trans. Biomed. Eng. 53 (11), 2139–2147.
Whiteley, J., 2007. Physiology driven adaptivity for the numerical solution of the bidomain equations. Ann. Biomed. Eng. 35 (9), 1510–1520.
Whiteley, J., 2008. An efficient technique for the numerical solution of the bidomain equations. Ann. Biomed. Eng. 36 (8), 1398–1408.

RECENT REPORTS

34/10	A random projection method for sharp phase boundaries in lattice Boltzmann simulations	Reis Dellar
35/10	Regularized Particle Filter with Langevin Resampling Step	Duan Farmer Moroz
36/10	Sequential Inverse Problems Bayesian Principles and the Logistic Map Example	Duan Farmer Moroz
37/10	Circumferential buckling instability of a growing cylindrical tube	Moulton Goriely
38/10	Preconditioners for state constrained optimal control problems with Moreau-Yosida penalty function	Stoll Wathen
39/10	Local synaptic signaling enhances the stochastic transport of motor-driven cargo in neurons	Newby Bressloff
40/10	Convection and Heat Transfer in Layered Sloping Warm-Water Aquifer	McKibbin Hale Style Walters
41/10	Optimal Error Estimates of a Mixed Finite Element Method for Parabolic Integro-Differential Equations with Non Smooth Initial Data	Goswami Pani Yadav
42/10	On the Linear Stability of the Fifth-Order WENO Discretization	Motamed Macdonald Ruuth
43/10	Four Bugs on a Rectangle	Chapman Lottes Trefethen
44/10	Mud peeling and horizontal crack formation in drying clay	Style Peppin Cocks
45/10	Binocular Rivalry in a Competitive Neural Network with Synaptic Depression	Kilpatrick Bressloff
46/10	A theory for the alignment of cortical feature maps during development	Bressloff Oster
47/10	All-at-Once Solution if Time-Dependent PDE-Constrained Optimisation Problmes	Stoll Wathen
48/10	Possible role of differential growth in airway wall remodeling in asthma	Moulton Goriely

49/10	Variational Data Assimilation Using Targetted Random Walks	Cotter Dashti Robinson Stuart
50/10	A model for the anisotropic response of fibrous soft tissues using six discrete fibre bundles	Flynn Rubin Nielsen
51/10	STOCHSIMGPU Parallel stochastic simulation for the Systems Biology Toolbox 2 for MATLAB	Klingbeil Erban Giles Maini
52/10	Order parameters in the Landau-de Gennes theory - the static and dynamic scenarios	Majumdar
53/10	Liquid Crystal Theory and Modelling Discussion Meeting	Majumdar Mottram
54/10	Modeling the growth of multicellular cancer spheroids in a bioengineered 3D microenvironment and their treatment with an anticancer drug	Loessner Flegg Byrne Hall Moroney Clements Hutmacher McElwain
55/10	Scalar Z, ZK, KZK, and KP equations for shear waves in incompressible solids	Destrade Goriely Saccomandi
56/10	The Influence of Bioreactor Geometry and the Mechanical Environment on Engineered Tissues	Osborne ODea Whiteley Byrne Waters

Copies of these, and any other OCCAM reports can be obtained from:

Oxford Centre for Collaborative Applied Mathematics
Mathematical Institute
24 - 29 St Giles'
Oxford
OX1 3LB
England
www.maths.ox.ac.uk/occam