NUMERICAL LINEAR ALGEBRA WITH APPLICATIONS
Numer. Linear Algebra Appl. 2004; **11**:261–277 (DOI: 10.1002/nla.381)

A fully implicit parallel algorithm for simulating the non-linear electrical activity of the heart

Maria Murillo[‡] and Xiao-Chuan Cai[*,†]

Department of Computer Science, University of Colorado at Boulder, Boulder, CO 80309, U.S.A.

SUMMARY

In this paper, we study a fully implicit parallel Newton–Krylov–Schwarz method (NKS) for solving the bidomain equations describing the electrical excitation process of the heart. NKS has been used successfully for many non-linear problems, but this is the first attempt to use this method for the bidomain model which consists of a system of time dependent partial differential equations of mixed type. Our experiments on parallel computers show that the method is scalable and robust with respect to many of the parameters in the bidomain model. In the outer layer of the algorithm, we use a non-linearly implicit backward Euler method to discretize the time derivative, and the resulting systems of large sparse non-linear equations are solved using an inexact Newton method. The Jacobian system required to solve in each Newton iteration is solved with a GMRES method preconditioned by a new component-wise restricted additive Schwarz preconditioner. The efficiency and robustness of the overall method depend heavily on what preconditioner we use. By comparing several preconditioners, we found our new restricted additive Schwarz method offers the best performance. Our parallel software is developed using the PETSc package of Argonne National Laboratory. Numerical results obtained on an IBM SP will be reported. Copyright © 2004 John Wiley & Sons, Ltd.

KEY WORDS: bidomain model; implicit method; Newton–Krylov–Schwarz algorithm; parallel computing

1. INTRODUCTION

Understanding the electrical mechanism in the heart is crucial in the cardiac research. As computing technologies advance, numerical simulation is becoming an increasingly useful tool for researchers. In this paper, we study a parallel and fully implicit computational method for solving the system of partial differential equations (PDE) that describes the electrical activity of the heart. Several PDE based models exist for such kind of simulations, we will focus on the most popular bidomain model, which assumes that the heart tissue consists of two domains: the intracellular and the extracellular domains that are separated by a membrane. The

*Correspondence to: X.-C. Cai, Department of Computer Science, University of Colorado at Boulder, Boulder, CO 80309, U.S.A.
†E-mail: cai@cs.colorado.edu
‡E-mail: murillo@cs.colorado.edu

Contract/grant sponsor: National Science Foundation; contract/grant number: ACI-0072089, ACI-0305666

Received 15 May 2003
Accepted 15 December 2003

changes of the electrical potential across the different domains generate the electrical waves that propagate through the cells. The bidomain model consists of a system of time dependent non-linear partial differential equations of mixed type. To solve the system, we investigate a Newton–Krylov–Schwarz (NKS) based non-linearly implicit technique on parallel computers. We are able to achieve good convergence even when the time step size is much larger than previously used by other researchers. We also show that the method scales well with large number of processors. Below we give a brief overview of the research in the modelling of the electrical system of the heart.

In 1950, Hodgkin and Huxley [1] gave the first description of a mathematical model of axon nerve cells, which serves as the basis for all the current models for the electrical system of the heart. The basic model, known as the fibre model, assumes that the tissue is nothing more than a single fibre (or cable), and it is associated with the well-known cable equation. This simple equation is a parabolic equation. In one-dimensional space, it can be written as

$$\frac{\partial V}{\partial t} = \frac{\partial^2 V}{\partial x^2} + f(V)$$

where V refers to the voltage of the electrical current, and f is some function of V representing the ionic current. Later, the model was extended to include three distinct areas within the heart tissue: the intracellular domain, the extracellular domain, and a membrane that separates the two domains. This is the so-called bidomain model, which is the most accepted model today, with some slight variations to its complexity. The electrical activity of the heart originates because of the changes in potential across these domains, which in turn, generate waveforms that propagate through the different cells [2].

The bidomain model [3] can be expressed as a system of reaction–diffusion equations consisting of a non-linear parabolic equation in V coupled with an elliptic equation in the extracellular potential U, and is given as

$$\chi c_m \frac{\partial V}{\partial t} + I_{\text{ion}} - I_{\text{app}} = -\nabla \cdot (D_e \nabla U)$$

$$\nabla \cdot (D_i \nabla V) = -\nabla \cdot (D \nabla U)$$

where χ and c_m are constants, I_{ion} is the ionic current, and I_{app} is any current applied to the system. The matrices D_e, D_i refer to the conductivity tensors. The ionic current can alter the complexity of the bidomain model considerably. Several authors have attempted to describe this. The more complex ones are based on the Hodgkin–Huxley formulation, or its variations, which are summarized in Reference [4] and references therein. One attempt to simplify the Hodgkin–Huxley approach leads to the development of the FitzHugh–Nagumo equations [5], which gives a more simpler version of the bidomain equations; for example, in the case of a medium with proportional conductivities, we have

$$\frac{\partial V}{\partial t} = \nabla \cdot (D \nabla V) + cV(V-1)(\alpha - V) - w$$

$$\frac{\partial w}{\partial t} = \varepsilon(V - \gamma w)$$

where c, α, ε, and γ are membrane constants, and w is the recovery potential. Although this is a much simpler mathematical model, it can still show some effect, such as the repolarization, which appears during the recovery phase or after the wavefront has passed through a section.

In our implementation, we use the bidomain equations as just introduced. The conductivity tensors are assumed to be diagonal arrays, and for the ionic current we use the FitzHugh–Nagumo formulation which allows the equations to be expressed as

$$\chi c_m \frac{\partial V}{\partial t} + cV(V-1)(\alpha - V) - w - I_{\text{app}} = -\nabla \cdot (D_e \nabla U)$$

$$\nabla \cdot (D_i \nabla V) = -\nabla \cdot (D \nabla U)$$

$$\frac{\partial w}{\partial t} = \varepsilon(V - \gamma w) \tag{1}$$

where $c, \alpha, \varepsilon, \gamma$ are membrane constants, and w is the recovery potential variable. For simplicity, we assume the model is defined on a simple two-dimensional domain $\Omega = (0,1) \times (0,1)$. In our experiments, we use Neumann boundary conditions which assume that there is no conductivity going out of the heart; i.e. sealed boundaries.

Another important issue when determining an appropriate model for the electrical activity of the heart is the conductivity tensor D. Depending on the structure of this matrix, we are incorporating various degrees of anisotropy to the system. The simplest one is a diagonal matrix, which indicates that the conductivity varies only along the co-ordinate system. A more complex system requires the inclusion of a full matrix where the conductivity varies in every direction, or even the most complex one represented by a full tensor. An associated element is the fact that in the bidomain model, we have different degrees of anisotropy in the intracellular and extracellular media.

There have been numerous attempts to numerically solve the heart excitation problem. In Reference [6], Hooke *et al.* present a general version of algebraic transformations that have been applied to the bidomain equations. For the most part, the explicit Euler's method is the choice for the time integration. In most cases, iterative methods (such as SOR) are used for solving the resulting equations, as is in Reference [7].

Veronese and Othmer [8] used a split-step method, which is a hybrid of an alternating direction implicit scheme for the non-linear parabolic equation, and a multigrid approach for the linear elliptic equation (for the transmembrane potential). Bogar [4] extended the use of a semi-implicit method for the bidomain model, which had only been used for monodomain models. This approach splits the non-linear term into two parts: one is resolved implicitly, the other, explicitly. For anisotropic bidomain models, simulations were run on 65×65 grids, with a total of 25 ms of propagation time, and time step of 0.1 ms. The resulting total CPU time was 66.26 s. Pennacchio and Simoncini [9] studied the convergence of various linear iterative solvers that were applied to the bidomain model at just one instant of time. They looked at various combinations of block Gauss–Seidel, and variations of preconditioned CG, with SOR and incomplete factorization as the preconditioner. Their estimated time for simulating about 40 ms of the excitation, with a time step size of 0.04 ms, on a $60 \times 60 \times 18$ mesh, is about 3 h.

Because of the complexities of implicit methods, researchers have made few attempts to use them. Stalidis *et al.* [10] used an implicit finite difference method to the monodomain model on a grid of 50×50 cells. Thakor and Fishler [11] used larger grids (200×250 cells). Hooke *et al.* [6] attempted to use an implicit scheme with Newton's method, using the GMRES method to solve the corresponding linear system of equations at each Newton step. They found their methodology to be ineffective or lacking efficiency for the bidomain problem, however it worked well in monodomain cases. Pormann [12] incorporated a similar approach using a matrix splitting procedure that resulted in a complicated set of steps where individual portions of the Jacobian matrix were inverted. They computed the solutions of these individual systems using the conjugate gradient method or GMRES, with some sequential preconditioners, such as SOR or incomplete LU factorizations. Lines *et al.* [13, 21] studied an operator splitting approach, combined with finite element and the Runge–Kutta method for the integration of the ODE system representing the ionic current. They carried out the process in a sequential manner; first, they integrated the ODE corresponding to the ionic current, then they solved each equation of the bidomain model in a cascade fashion (first, the parabolic equation in V, and then the elliptic equation in U_e). In solving the reaction–diffusion for V, they used the explicit method, and the problem became linear.

The rest of the paper is organized as follows. In Section 2, we present the details of the bidomain model, its discretization, and the numerical methods we use to solve the discrete system. The simulation results and the parallel performance of the numerical algorithm and software are discussed in Section 3. We make several final remarks in Section 4, which concludes the paper.

2. NUMERICAL METHODS AND PARALLEL IMPLEMENTATION

2.1. Discretization of the bidomain model

In this paper, we assume the bidomain model is defined on $\Omega = (0,1) \times (0,1)$, which is covered by a uniform grid. $\Delta x = \Delta y$ are mesh sizes in the x and y directions, respectively. Let m be the total number of grid points in either the x or y direction. Δt is the time step size. To discretize equation (1) we use the usual 5-point central finite difference method for the space variables and the fully implicit backward Euler for the time variable. The resulting set of sparse, non-linear algebraic system of equations is solved using the NKS iterative method [14] at every time step.

We first re-write (1) as follows:

$$c_m \frac{\partial V}{\partial t} - \sigma_l^i \frac{\partial^2 V}{\partial x^2} - \sigma_t^i \frac{\partial^2 V}{\partial y^2} + cV(\alpha - V)(V - 1) - w = \sigma_l^i \frac{\partial^2 U_e}{\partial x^2} + \sigma_t^i \frac{\partial^2 U_e}{\partial y^2} + I_{\text{app}}$$

$$-(\sigma_l^i + \sigma_l^e) \frac{\partial^2 U_e}{\partial x^2} - (\sigma_t^i + \sigma_t^e) \frac{\partial^2 U_e}{\partial y^2} = \sigma_i^l \frac{\partial^2 V}{\partial x^2} + \sigma_i^t \frac{\partial^2 V}{\partial y^2}$$

$$\frac{\partial w}{\partial t} = \varepsilon(V - \gamma w)$$

To simplify the notation we let $U := U_e$. The fully discretized version of the above system take the form

$$
\begin{aligned}
& c_m \frac{V_{i,j}^n - V_{i,j}^{n-1}}{\Delta t} - \sigma_l^i \frac{V_{i-1,j}^n - 2V_{i,j}^n + V_{i+1,j}^n}{(\Delta x)^2} - \sigma_t^i \frac{V_{i,j-1}^n - 2V_{i,j}^n + V_{i,j+1}^n}{(\Delta y)^2} \\
& \quad + cV_{i,j}^n (V_{i,j}^n - 1)(\alpha - V_{i,j}^n) - w_{i,j}^n \\
& \quad = \sigma_l^i \frac{U_{i-1,j}^n - 2U_{i,j}^n + U_{i+1,j}^n}{(\Delta x)^2} + \sigma_t^i \frac{U_{i,j-1}^n - 2U_{i,j}^n + U_{i,j+1}^n}{(\Delta y)^2} \\
& \qquad + I_{\mathrm{app}_{i,j}}^n \\
& \qquad - (\sigma_l^i + \sigma_l^e)(U_{i-1,j}^n - 2U_{i,j}^n + U_{i+1,j}^n) - (\sigma_t^i + \sigma_t^e)(U_{i,j-1}^n - 2U_{i,j}^n + U_{i,j+1}^n) \\
& \quad = \sigma_l^i (V_{i-1,j}^n - 2V_{i,j}^n + V_{i+1,j}^n) + \sigma_t^i (V_{i,j-1}^n - 2V_{i,j}^n + V_{i,j+1}^n)
\end{aligned}
$$

and

$$
\frac{w_{i,j}^n - w_{i,j}^{n-1}}{\Delta t} = \varepsilon (V_{i,j}^n - \gamma w_{i,j}^n)
$$

For the initial condition, we assume our system is at rest at the initial time ($t = 0$). As indicated in Reference [9], both the resting potentials and w are zero. Also, in order to excite the system, we apply a current I_{app} of 40 amp/cm^2 for a period of 1.0 ms, to the centre portion of our domain. Other authors, such as [4, 15] have used the value in their numerical experiments.

For the bidomain model, it is commonly assumed that there is no flux across the boundaries [4]. In other words, because the potential V has a constant value before and behind the exciting front, we can assume that it is not affected by the outside medium [3]. In particular, for our model, these boundary conditions are expressed as $\partial V/\partial n = 0$ and $\partial U/\partial n = 0$. In the discretization, we use

$$
\left.
\begin{array}{ll}
(\textit{lower edge}, y = 0), & \begin{cases} F(V) = V_{i,0} - V_{i,1}, \\ F(U) = U_{i,0} - U_{i,1}, \end{cases} \\
(\textit{upper edge}, y = 1), & \begin{cases} F(V) = V_{i,m} - V_{i,m-1}, \\ F(U) = U_{i,m} - U_{i,m-1}, \end{cases}
\end{array}
\right\} \quad 0 < i < m
$$

$$
\left.
\begin{array}{ll}
(\textit{left edge}, \ x = 0), & \begin{cases} F(V) = V_{0,j} - V_{1,j}, \\ F(U) = U_{0,j} - U_{1,j}, \end{cases} \\
(\textit{right edge}, x = 1), & \begin{cases} F(V) = V_{m,j} - V_{m-1,j}, \\ F(U) = U_{m,j} - U_{m-1,j}, \end{cases}
\end{array}
\right\} \quad 0 < j < m
$$

In most of the existing numerical methods, for example Reference [13], for solving the above finite difference equations, the unknowns are ordered in terms of the physical variables

separately to form the following, more compact, system of equations:

$$\begin{aligned}\mathbf{F}_V &= \mathbf{V}^n + \Delta t\tilde{\mathbf{F}}_V(V^n, U^n, w^n) - \mathbf{V}^{n-1}\\ \mathbf{F}_U &= \tilde{\mathbf{F}}_U(V^n, U^n, w^n)\\ \mathbf{F}_w &= \mathbf{w}^n - \Delta t\tilde{\mathbf{F}}_w(V^n, U^n, w^n) - \mathbf{w}^{n-1}\end{aligned} \tag{2}$$

In our work, we find the interlacing ordering performs much better. More precisely speaking, to write the above finite difference system into a system of algebraic equations, we order the unknowns by the mesh points as follows:

$$\mathbf{G} = [(V_{11}, U_{11}, w_{11}), (V_{21}, U_{21}, w_{21}), \ldots, (V_{mm}, U_{mm}, w_{mm})]^{\mathrm{T}}$$

The corresponding equations are ordered in exactly the same way, and the system is denoted by

$$F(G) = 0 \tag{3}$$

For simplicity, we ignored the sub- and super-scripts.

2.2. Solving the algebraic systems

To find G at the nth time step, a non-linear system of the form (3) needs to be solved. Using G at the $(n-1)$th time step as the initial guess, we use an inexact Newton method [16] to solve (3). Let $J = F'$ be the Jacobian matrix, which is calculated analytically in this paper. The two major steps of the inexact Newton's method are

Find ΔG^k that satisfies

$$\|J(G^k)\Delta G^k + F(G^k)\| \leqslant \eta\|F(G^k)\|$$

for any $\eta < 1$, and then set

$$G^{k+1} = G^k - \Delta G^k \tag{4}$$

The inexact Newton method presented above requires that we solve (inexactly) the following linear system of equations:

$$J(G^k)\Delta G^k = -F(G^k)$$

A restricted additive Schwarz preconditioned GMRES method is used for solving such a linear system. In particular, we use GMRES [17] for the following system

$$M^{-1}J\Delta G = -M^{-1}F \tag{5}$$

where M^{-1} is a parallel preconditioner. Several choices of M^{-1} are available in PETSc [18], including the block Jacobi preconditioners, the regular additive Schwarz [19] and the restricted additive Schwarz preconditioners (RAS) [20]. All of the preconditioner are *algebraic* in the sense that the multi-component nature of our linear system is not taken into account. In this work, we introduce a component-wise restricted additive Schwarz method, which is a simple

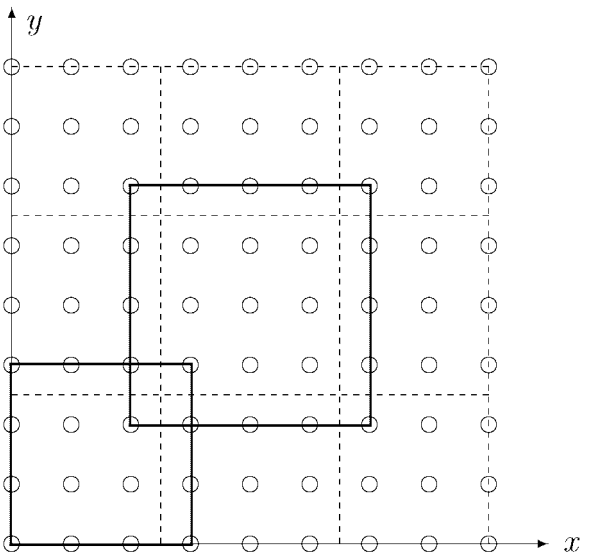

Figure 1. A sample 9×9 fine mesh with a 3×3 subdomain partition. The 'o' are the mesh points. The dashed lines indicate a $3 \times 3 = 9$ subdomain non-overlapping partitioning. The solid lines indicate the subdomains with $\delta = 1$.

application of the algebraic RAS idea [20] for each of the three variables (U, V, w) of the PDE separately.

In order to define a parallel preconditioner M^{-1}, we need to partition the unknowns of the algebraic system and map them onto different processors. The standard partitioning techniques in PETSc are all based on the partitioning of the *matrix* itself, and the quality of the partitioning, as well as the quality of the resulting block preconditioner, depends on how the mesh points are ordered and how the multiple unknowns at each mesh point are ordered. Our experiences show that a better partitioning can often be obtained by looking at the *mesh*, not the matrix, especially for systems arising from the discretization of multi-component PDEs.

To obtain the overlapping partition, we first partition the domain into non-overlapping subdomains Ω_i^0, $i = 1, \ldots, N$. Then we extend each subdomain Ω_i^0 to a larger subdomain Ω_i^δ, i.e. $\Omega_i^0 \subset \Omega_i^\delta$. Here δ is an integer indicating the size of the overlap. Only simple box decomposition is considered in this paper; i.e. all the subdomains Ω_i^0 and Ω_i^δ are rectangular and made up of integral numbers of fine mesh cells. For boundary subdomains, we simply cut off the part that is outside Ω. A sample mesh with an overlapping partition is shown in Figure 1. Let n be the number of mesh points in Ω, n_i the number of mesh points in Ω_i^0, and n_i^δ the number of mesh points in Ω_i^δ. For each subdomain Ω_i^0, we define I_i^0 as an $n_i^0 \times n_i^0$ block sub-identity matrix whose diagonal element, $(I_i^0)_{k,k}$, is either an 3×3 identity matrix if the mesh point $x_k \in \Omega_i^0$ or an 3×3 zero matrix if x_k is outside of Ω_i^0. Similarly, we introduce a block sub-identity matrix $(I_i^\delta)_{k,k}$ for each Ω_i^δ. Here, δ is the overlapping size. With the subdomain restriction operator I_i^δ, we define the subdomain Jacobian matrix

$$J_i = I_i^\delta J I_i^\delta$$

Note that although J_i is not invertible, we can invert its restriction to the subspace

$$J_i^{-1} \equiv ((J_i)_{|L_i})^{-1}$$

where L_i is the vector space spanned by the unit vectors defined on Ω_i^δ (i.e., if the kth component of the unit vector is equal to 1, then the corresponding mesh point x_k must be in Ω_i^δ). Recall that the regular AS preconditioner is defined as [19]

$$M_{\mathrm{AS}}^{-1} = I_1^\delta J_1^{-1} I_1^\delta + \cdots + I_N^\delta J_N^{-1} I_N^\delta$$

Our new RAS preconditioner can simply be described as follows:

$$M_{\mathrm{RAS}}^{-1} = I_1^0 J_1^{-1} I_1^\delta + \cdots + I_N^0 J_N^{-1} I_N^\delta$$

One can also define the so-called 'interpolate RAS' as follows:

$$M_{\mathrm{RAS}}^{-1} = I_1^\delta J_1^{-1} I_1^0 + \cdots + I_N^\delta J_N^{-1} I_N^0$$

The performance of the two RAS preconditioners are nearly identical for our problem.

We remark that the matrix–vector multiply $I_i^\delta G$ does not involve any arithmetic operations, but does involve communications among neighbouring processors. However, $I_i^0 G$ involves neither computation nor communication. This makes the RAS preconditioner much cheaper than the AS preconditioner. We will show numerically that using the RAS preconditioner can also reduce the number of GMRES iterations. For single component problems, the new RAS preconditioners are the same as what was introduced in Reference [20], but for multi-component problems, the new RAS preconditioners are less algebraic, and can not be defined only based on the given sparse matrix.

3. SIMULATION RESULTS AND PARALLEL PERFORMANCE

In this section, we present some numerical results for solving the bidomain model equations. We make extensive use of the software package PETSc of the Argonne National Laboratory [18]. All results are obtained from experiments ran on the NPACI's IBM-SP machine at the San Diego Supercomputing Center.

3.1. Parameters in the bidomain model

For the conductivity parameters we use the so-called 'nominal' values because of the general understanding that these are probably the values present in the actual heart tissue [15]. For the ionic current we use the FitzHugh–Nagumo model. These parameters (as given in Reference [4]) are given in Tables I and II.

Table I. Physical parameters for the bidomain model.

$\chi = 2000\ \mathrm{cm}^{-1}$	$c_m = 1.0\ \mu\mathrm{F\ cm}^{-2}$
$\sigma_e^l = 4.0 \times 10^{-3}\ \Omega^{-1}\ \mathrm{cm}^{-1}$	$\sigma_e^t = 4.0 \times 10^{-3}\ \Omega^{-1}\ \mathrm{cm}^{-1}$
$\sigma_i^l = 4.0 \times 10^{-3}\ \Omega^{-1}\ \mathrm{cm}^{-1}$	$\sigma_i^t = 1.0 \times 10^{-3}\ \Omega^{-1}\ \mathrm{cm}^{-1}$

Table II. FitzHugh–Nagumo parameters for the ionic current.

$c = 2.0$	$\alpha = 0.1$
$\gamma = 0.5$	$\varepsilon = 0.002$

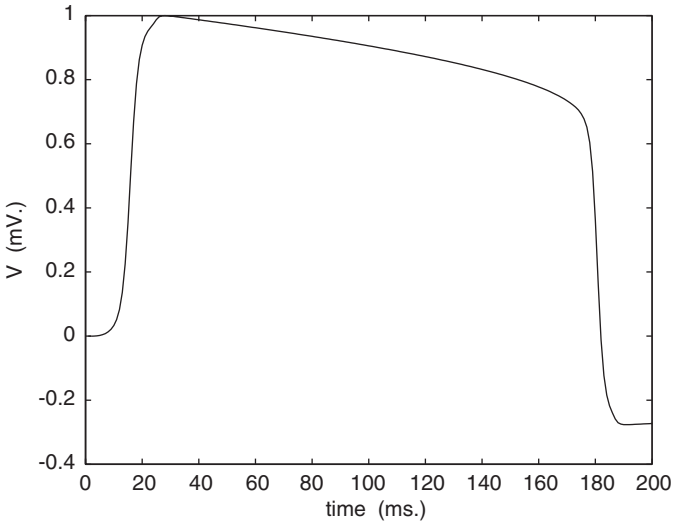

Figure 2. The computed AP signal. Depolarization, plateau, and repolarization phases are present.

The computed AP signal is presented in Figure 2. The observed AP signal remain the same among the various experiments performed using different number of processors. Here, we can see the three distinctive phases: depolarization, plateau, and repolarization. We have allowed some degree of anisotropy in our media, as expressed by the different conductivity coefficients used. In Figure 3, we can observe how along the axis where the conductivity is larger, the activation has spread faster than in the other direction. The picture shows in particular the activation process after 10 ms of simulation; once the whole area is activated, the phenomenon would not be observed.

3.2. Performance of the linear and non-linear solvers

To obtain a time accurate solution, we use $\Delta t = 1$ ms for all the numerical experiments. For the non-linear solver we use the following stopping conditions: $\|F(G^k)\| \leqslant 10^{-4}\|F(G^0)\|$, and in the linear solver we set $\eta = 10^{-6}$.

The performance of the inexact Newton method is shown in Figure 4. In the figure, we show that the total number of non-linear iterations per time step. The numbers are quite small. In all cases, the number of non-linear iterations remains nearly constant at three during the plateau phase, and grows only to about 4 or 5 iterations during the repolarization and depolarization phases. In some instances of time, we observed a maximum total of six or seven iterations. We also observe that these numbers remain nearly unchanged independently

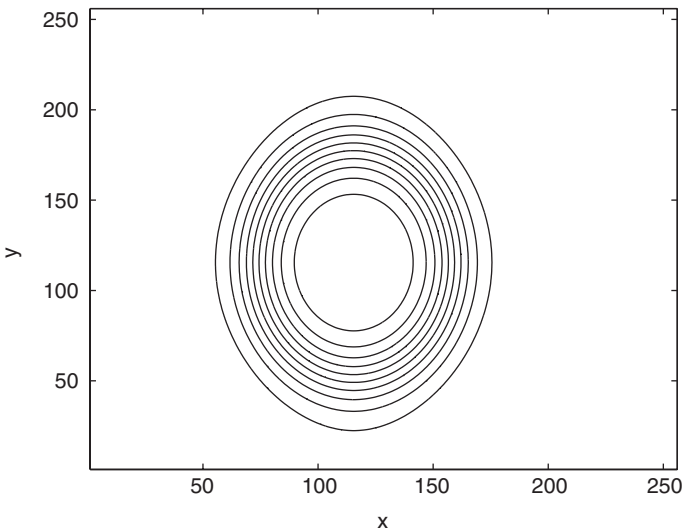

Figure 3. The figure shows the transmembrane potential (V) after it has propagated for 10 ms.

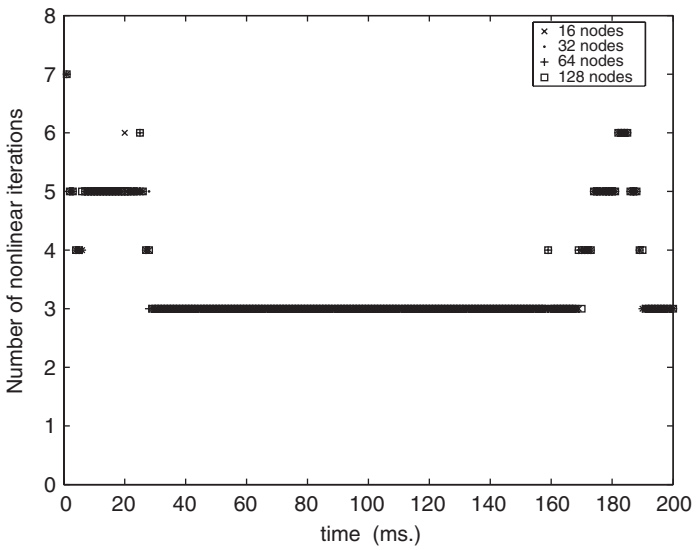

Figure 4. Non-linear iterations per time step, for a 512×512 mesh. The number of processors vary from 16 to 128.

of the mesh size, and the number of processors. In Figure 5, we present the non-linear residual for the case where we use 16 processors and several different mesh sizes. We can see in these results that the variations in all these curves are small, and therefore the non-linear residual seems to converge equally independently of the size of the problem.

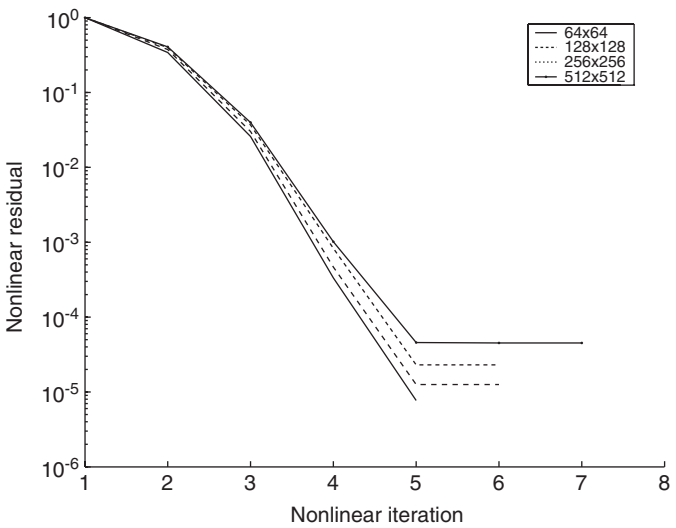

Figure 5. Non-linear residual for various grid sizes, using 16 processors and at time = 20 ms (depolarization phase).

Another interesting feature we observed with respect to the non-linear iterations within our method, was the average number of linear iterations that are required per each non-linear iteration. If we first look at a given problem size, and allow the number of processors to vary, then we can observe that this average of linear of iterations is relatively constant as shown in Figure 6. We can also observe that this number of linear iterations decreases somewhat rapidly as the non-linear iterations progress, showing a relatively larger number only on the first two or three non-linear iterations.

In the case where we look at a particular number of processors, but allow the problem size to vary, we see that the pattern of decrease in the number of linear iterations from one non-linear iteration to the next is somewhat similar, as shown in Figure 7. However, we can appreciate a considerable difference in the actual numbers, where the problems with larger sizes require much more linear iterations for at least the first three non-linear iterations. This indicates that the number of linear iterations have a clear dependence on the size of the problem, as opposed to the case of non-linear iterations where we saw no clear variation.

In Figure 8, we compare the three additive Schwarz type preconditioners. In both the restricted versions, the values from neighbouring subdomains, or neighbouring processors, are communicated at only one of the stages of the matrix–vector multiplications (either during restriction or during interpolation, but not both). It is clear, from Figure 8, the two restricted versions are very similar in terms of the number of GMRES iterations per time step. However, when we compare with regular additive Schwarz method, where values from other processors are incorporated at both ends (restriction and interpolation), we observed a considerable increase on the number of linear iterations. The phenomenon is considerably more pronounced during the depolarization and repolarization phases. Because of these results, in the remainder of the experiments we will only use the RAS preconditioner.

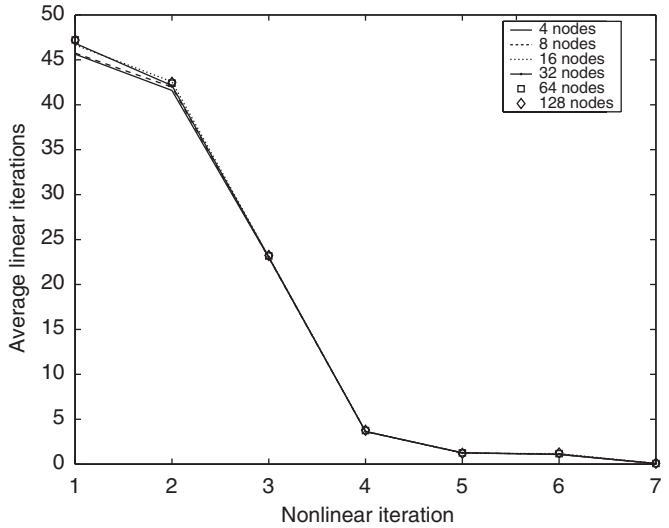

Figure 6. Average number of linear iterations, varying the number of processors on a 256×256 grid.

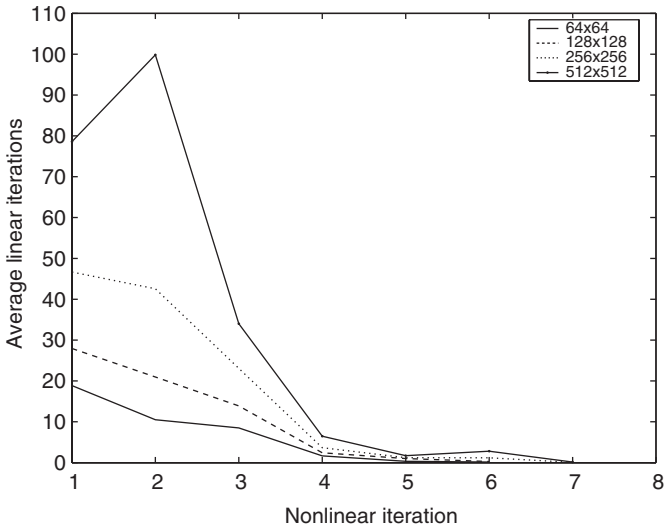

Figure 7. Average number of linear iterations, varying the grid or problem size.

When using Schwarz type preconditioners, the number of GMRES iterations often depends, to a certain degree, on the overlapping factor. This has been studied for many different equations. In the following experiment, we fix the mesh to 128×128 and the number of processors to 16. The total number of GMRES iterations are given in Figure 9 for three different overlapping sizes, 1, 4, and 20. It is a bit surprising that the number of iterations is nearly independent of the overlapping size. In the rest of the experiments, we use overlap $= 1$.

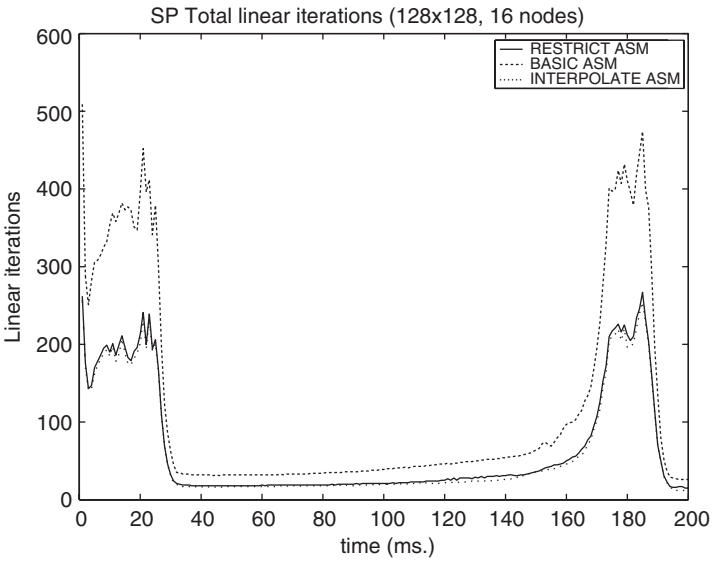

Figure 8. Compare different additive Schwarz preconditioners. The figure shows the total number of GMRES iterations per time step on a 128×128 mesh and 16 processors.

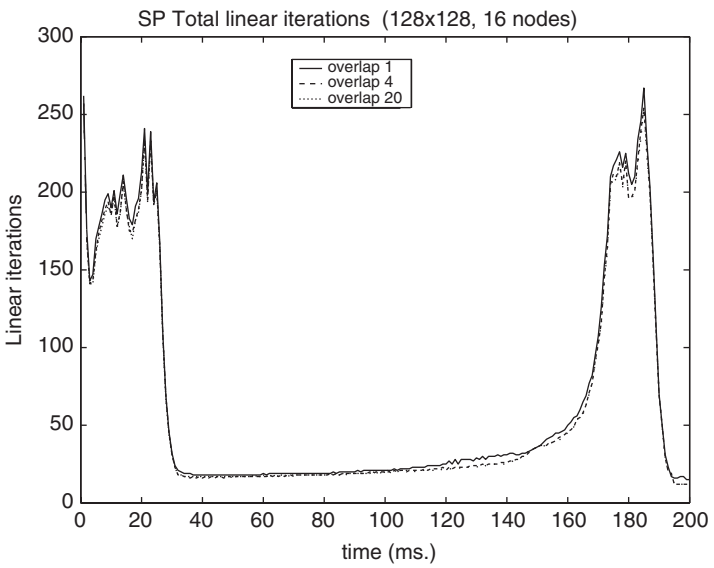

Figure 9. Varying the overlapping size. 16 processors and 128×128 mesh.

To see the scalability of the preconditioner with respect to the mesh size, in Figure 10, we show the total number of GMRES iterations per time step for four different mesh sizes, from 64×64 to 512×512. We fix the number of subdomains, or the number of processors, to 16.

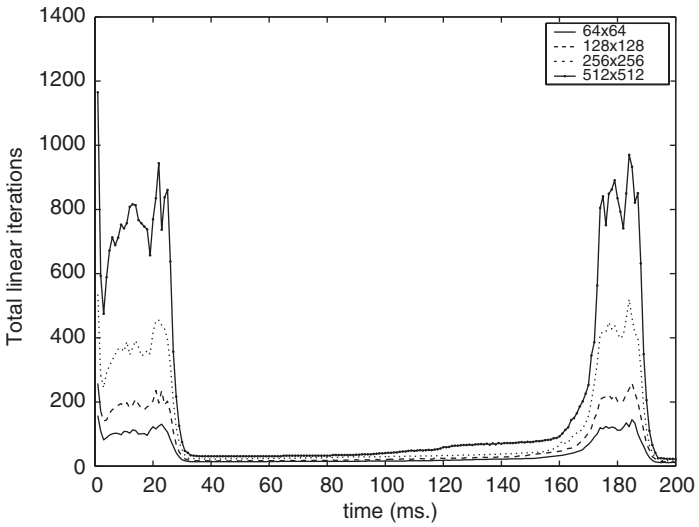

Figure 10. The total number of GMRES iterations per time step with varying mesh sizes. 16 processors are used in all the calculations.

The subdomain problems are solved inexactly with ILU(0). We observe that the numbers increase as we refine the mesh. This indicates that the single level RAS preconditioned GMRES is not scalable, in terms of the number of iterations. A coarse grid is probably necessary in order to have a nearly constant number of iterations. However, as will be shown later in this paper, a CPU based scalability can still be obtained even without a coarse grid.

On the other hand, if we maintain the mesh size as a constant but vary the number of processors or nodes used, then we can observe that the pattern for the total number of linear iterations per time step is virtually identical in all cases, as presented in Figure 11. Only small, almost imperceptible, variations take place, and we can only observe these at the depolarization and repolarization phases. This indicates that there is no dependency between the number of linear iterations and the number of processors.

3.3. CPU performance

Table III summarizes the CPU times for different problem sizes and different number of processors (or nodes). The number of mesh points per processor can be calculated from the given information in Table III, since the total number of points and the total number of processors are both given, and the partition is uniform. As expected, the CPU times increase accordingly as we increase the problem size. For a given problem size, the CPU times decrease as we increase the number of processors.

These CPU timing results are better understood in terms of Figure 12, in which we show the actual values as connected lines, whereas the expected values for optimal scalability are shown as individual points. We observe that for the smaller problems, the scalability degrades quickly as we increase the number of processors. However, for the largest problem, we see a superlinear curve. In other words, our CPU time results are better than expected. This is likely due to the better cache performance of the algorithm/software.

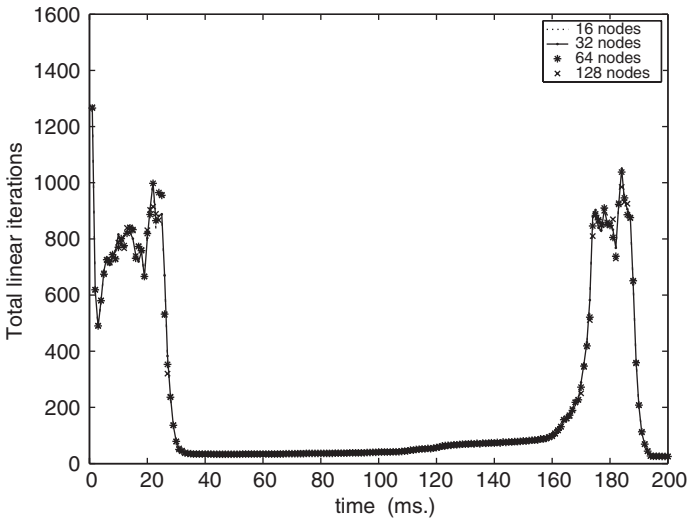

Figure 11. Total number of linear iterations on a fix 512×512 mesh with varying number of processors.

Table III. CPU time (s) for solving bidomain model using different mesh sizes and number of processors. '$*$': the mesh is either too small or too large for the given number of processors.

Processors	64×64	128×128	256×256	512×512
4	69	432	2837	$*$
8	44	270	1372	$*$
16	33	201	560	5472
32	28	165	261	2400
64	27	164	168	994
128	$*$	$*$	136	519

We observe the same type of behaviour by looking at Figure 13, where the speedups of various simulations are presented. In this figure, it is clear that for the largest mesh, the speedup is superlinear, which suggests that our algorithm and software implementation perform better if the problem size is sufficiently large. Even though one of the three components of the bidomain system (1) is elliptic, no coarse space seems necessary for the domain decomposition approach to be CPU time scalable.

4. FINAL REMARKS

In this paper, we study a non-linearly implicit method for solving the non-linear system of equations [4, 12] arising from the discretization of the bidomain equations modelling the electrical activation of the heart. We show numerically that the non-linearity appears in the bidomain model is weak, since the total number of non-linear iterations necessary for the convergence at each time step is relatively small, with a maximum of 6 or 7, and only

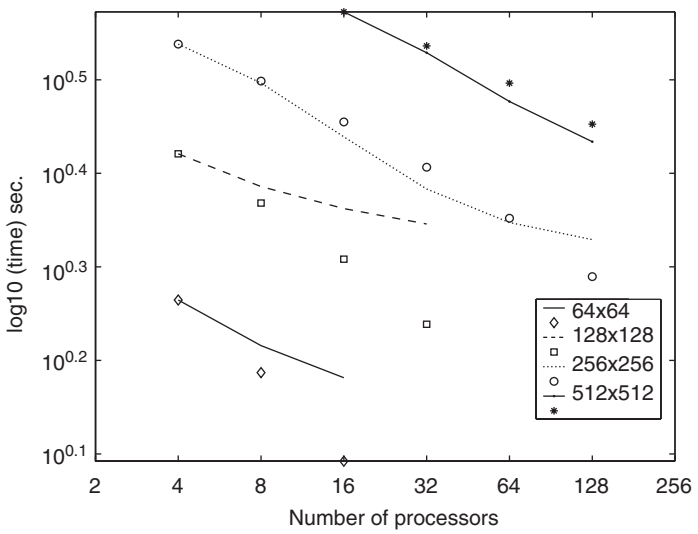

Figure 12. Total CPU time (s) for different mesh sizes and number of processors.

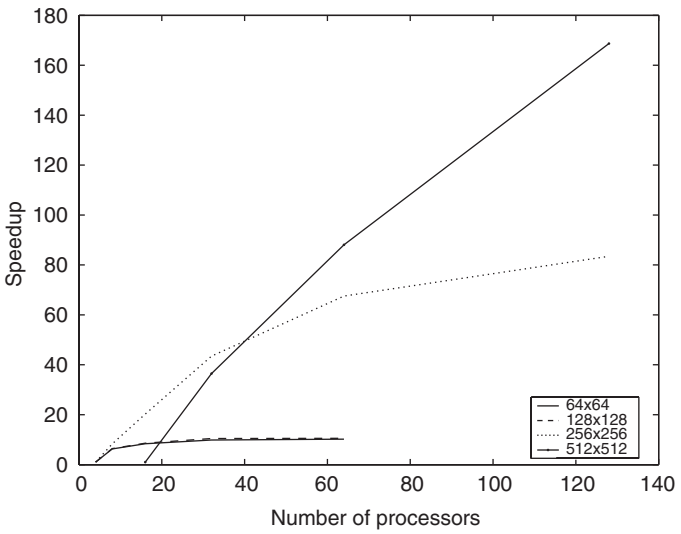

Figure 13. Fixed mesh size speedup.

at one or two instances of time during the activation process. We observe that only three non-linear iterations are required during the plateau phase, and four or five during the more complex repolarization and depolarization phases. Our numerical results also show that the number of non-linear iterations is independent of the size of the problem and the number of processors used to solve the problem. In short, the non-linear convergence is fast and perfectly scalable.

On the other hand, our results show that the total number of linear iterations depends on the size of the problem, although not affected by the number of processors. In other words, the single level RAS preconditioned GMRES is perfectly scalable in terms of the number of processors, but not in terms of the mesh sizes.

One of the most important results of this research is that we find that the overall algorithm is *CPU time* scalable if the mesh is fine enough. This suggests that the combination of the inexact Newton method with a single level RAS preconditioned GMRES is a good choice of this kind of problem. We are able to achieve superlinear CPU time scalability for large meshes and for large number of processors.

In summary, our experiments show that NKS methodology as applied to the bidomain equations is stable and capable of delivering accurate solutions. This approach allows us to use larger time step size than what is required by other methods. For example, our time step is one order of magnitude larger than what was used in Reference [4].

REFERENCES

1. Hodgkin A, Huxley A. A quantitative description of membrane current and its application to conduction and excitation in nerve. *Journal of Physiology* 1952; **117**:500–544.
2. Ruttkay-Nedecky I. Physiology of the cardiac electric field. In *Heart Function in Health and Disease*, Ostadal B, Dhalla N (eds). Kluwer Academic Publishing: Dordrecht, 1993; 111–125.
3. Grottum P, Tveito A, Zych D. Heart—numerical simulation of the excitation process in the human heart, *Technical Report* 1995 (see also http://www.oslo.sintef.no/adv/33/3340/diffpack).
4. Bogar K. A semi-implicit time scheme for the study of action potential propagation in the bidomain model. *Ph.D. Thesis*, University of Utah, 1999.
5. Boyett M, Clough A, Dekanski J, Holden A. Modeling cardiac excitation and excitability. In *Computational Biology of the Heart*, Panfilov A, Holden A (eds). Chapter 1. John Wiley and Sons: New York, 1997; 1–47.
6. Hooke N, Henriquez C, Lanzkron P, Rose D. Linear algebraic transformations of the bidomain equations: implications for numerical methods. *Mathematical Biosciences* 1994; **120**:127–145.
7. Latimer D, Roth B. Electrical stimulation of cardiac tissue by a bipolar electrode in a conductive bath. *IEEE Transactions on Biomedical Engineering* 1998; **45**:1449–1458.
8. Veronese S, Othmer H. *A Computational Study of Wave Propagation in a Model for Anisotropic Cardiac Ventricular Tissue*. Lecture Notes in Computer Science, vol. 919. Springer: Berlin, 1995; 248–253.
9. Pennacchio M, Simoncini V. Efficient algebraic solution of reaction–diffusion systems for the cardiac excitation process. *Journal of Computational and Applied Mathematics* 2002; **145**:49–70.
10. Stalidis G, Maglaveras N, Dimitriadis A, Pappas C, Strintzis M, Efstratiadis SN. Application of a 3-D ischemic heart model derived from MRI data to the simulation of the electrical activity of the heart. In *Computers in Cardiology*, Murray A, Arzbaecher R (eds). IEEE; New York, 1996; 329–332.
11. Thakor N, Fishler M. Initiation and termination of spiral waves in a two-dimensional bidomain model of cardiac tissue. In *Computers in Cardiology*, Murray A, Arzbaecher R (eds). IEEE; New York, 1996; 229–232.
12. Pormann J. A modular simulation system for the bidomain equations. *Ph.D. Thesis*, Duke University, 1999.
13. Lines G, Cai X, Tveito A. A parallel solution of the bidomain equations modeling the electrical activity of the heart. 2001, submitted (see also, http://www.simula.no/ glennli/heart/cvs1.pdf).
14. Cai X-C, Gropp WD, Keyes DE, Melvin RG, Young DP. Parallel Newton–Krylov–Schwarz algorithms for the transonic full potential equation. *SIAM Journal on Scientific Computing* 1998; **19**:246–265.
15. Hooke N. Efficient simulation of action potential propagation in a bidomain. *Ph.D. Thesis*, Duke University, 1992.
16. Eisenstat S, Walker H. Globally convergent inexact Newton methods. *SIAM Journal on Optimization* 1994; **4**:393–422.
17. Saad Y. *Iterative Methods for Sparse Linear Systems*. SIAM; Philadelphia, 2003.
18. Balay S, Buschelman K, Gropp WD, Kaushik D, Knepley M, McInnes LC, Smith BF, Zhang H. *PETSc Users Manual*, ANL-95/11—Revision 2.1.5, Argonne National Laboratory, 2002.
19. Dryja M, Widlund O. Domain decomposition algorithms with small overlap. *SIAM Journal on Scientific Computing* 1994; **15**:604–620.
20. Cai X-C, Sarkis M. A restricted additive Schwarz preconditioner for general sparse linear systems. *SIAM Journal on Scientific Computing* 1999; **21**:792–797.
21. Lines GT. Simulating the electrical activity of the heart. *Ph.D. Thesis*, University of Oslo, 1999.

