Full-Scale Simulation of Cardiac Electrophysiology on Parallel Computers

Xing Cai[1,2] and Glenn Terje Lines[1,2]

[1] Simula Research Laboratory, P.O. Box 134, N-1325 Lysaker, Norway
{xingca,glennli}@simula.no
[2] Department of Informatics, University of Oslo, P.O. Box 1080, Blindern, N-0316 Oslo, Norway

Summary. In this chapter, we will present an advanced parallel electro-cardiac simulator, which employs anisotropic and inhomogeneous conductivities in realistic three-dimensional geometries for modeling both the heart and the torso. Since partial differential equations (PDEs) constitute the main part of the mathematical model, this chapter thus demonstrates a concrete example of solving PDEs on parallel computers. It will be shown that good overall parallel performance relies on at least two factors. First, the serial numerical strategy must find a parallel substitute that is scalable with respect to both convergence and work amount. Second, care must be taken to avoid unnecessary duplicated local computations while maintaining an acceptable level of load balance.

1 Introduction

The contraction of the heart is triggered by a signal wave that propagates through the muscle tissue. Between two heart beats, each muscle cell is in a resting phase and has a surplus of negative charge compared to the exterior of the cell. This is called a negative transmembrane potential. When the signal wave reaches a cell, the conductivity properties of the cell membrane are altered, and positive ions are able to enter the cell. This is the so-called *depolarization* of the cell. This depolarization causes the neighboring cells to also alter their membrane conductance, thus allowing ions to enter. In this way the signal is sustained and travels through the whole heart. The aggregated effect of all the cells being depolarized is measurable on the body surface as the peak of the ECG signal.

The currently most accurate mathematical model of the electrical activities of the heart tissue is the Bidomain model, which will be defined precisely in Section 2. It consists of two coupled PDEs, where the primary unknowns are the electrical potentials outside and inside of the cell. The difficulty with solving this mathematical model is that the current crossing the membrane, hereafter referred to as the ionic current, depends nonlinearly on the potentials and also on a large number of other entities, such as ionic concentrations and permeability of the membrane. These entities are typically modeled by a system of ordinary differential equations (ODEs). A complete mathematical model thus consists of the Bidomain PDE system coupled with an ODE system. For an example of computer simulations of cardiac electrophysiology, we refer to Figure 1 which shows two snapshots from a prototypical three-dimensional simulation done in [19].

The ODE system can be quite complex. For example, the model of Winslow et al. [36] involves 30 state variables. As more biological data become available, more detailed descriptions are incorporated into the models, and consequently the model complexity increases. There also exist simplified models with only a few state variables. For some applications these simple models might be sufficient, but they are typically valid only within a narrow range. The complex models are more realistic as they include many of the subsystems within the cell, which may be calcium stored inside the cell or the state of the different membrane proteins. These proteins allow current to pass through the membrane, forming so-called ionic channels. Since the complex models are more realistic, they have a wider area of application. For example, one can investigate the effect of a drug that blocks an ionic canal by simply reducing the current going through this channel in a complex model. There is, in general, a complex interplay between different the ionic channels, and it is not easy to predict how the blocking of one type of channel affects the cell as a whole, or indeed the performance of the entire heart. Simulation of such systems will therefore give us an insight that is not easily obtainable in any other way.

The time scale of the biochemical process in a cell is very small compared with the duration of a heart beat. The depolarization of a cell takes about 1ms, in comparison with about one second for a heart beat. From a numerical point of view, another challenge is that the activation signal wave front is very narrow, about 1mm compared with 10cm for the whole heart. A commonly reported value for the required spatial resolution is 0.2mm, which is necessary for rendering the narrow wave front with sufficient accuracy; see e.g. [3, 33]. Using such a mesh point density over the entire heart volume of an adult means that about 40 million mesh points are required; see [18]. It is obvious that the computational challenges are formidable for such a numerical problem.

To deal with the computational burden, one often resort to studying simplified mathematical models. One option is to use the Monodomain model, which is a scalar PDE and can be derived as a special case of the Bidomain model. Another option is to use simpler models for the ionic current. These simplifications together can reduce the computation time by at least a factor of 10. This time saving factor is due to a combination of at least four reasons: a) the number of unknowns in the Bidomain model is twice of that in the Monodomain model, b) the resulting linear system of the Bidomain model (from applying a linearization and spatial discretizations) requires considerably more iterations of an iterative solver than the Monodomain model,

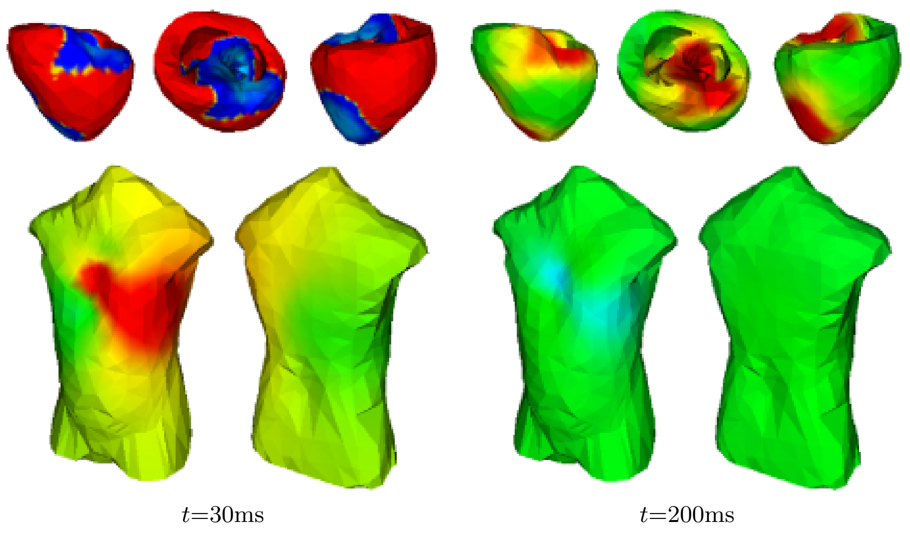

Fig. 1. Snapshots from two time levels of a simulation of the electrical field in the human heart and torso. At each time level, the electrical potential distribution on the heart surface is shown at three different angles, while the distribution on the torso surface is shown at two different angles.

especially in the absence of a powerful preconditioner (see e.g. [27]), c) the computational cost of one matrix-vector product associated with the Bidomain model is approximately four times of that with the Monodomain model, and d) a simpler ionic current model results in an ODE system that has fewer equations and is easier to solve. To avoid the need for 40 million mesh points one can also consider a volume smaller than the entire heart, or, alternatively, use a coarser mesh resolution. Both these approaches are popular since they make it easier to fit the whole problem onto a serial computer, but at the expense of less realistic results.

Performing full-scale simulations of the entire heart is only possible by using parallel computers. We can argue for this observation roughly as follows. Assuming 40×10^6 mesh points in the heart domain and using the Winslow ODE model, the total number of degrees of freedom amounts to $32 \times 40 \times 10^6$, where for each mesh point there are two PDE degrees of freedom and 30 ODE degrees of freedom. At any given time level, storing the values of all the degrees of freedom will require approximately 10GB memory using double precisions. The time stepping scheme typically needs to store the values for two consecutive time levels. Moreover, the data structure needed for storing the computational meshes and the matrix for the discretized PDEs will require considerably more memory. Clearly, such a huge demand of memory can not be met by any serial computer, in addition to the equally huge demand of simulation time.

Parallel electro-cardiac simulation is still a relatively new research topic. Publications on this particular topic have been sparse. This is due to a combination of the complicated mathematical model, the necessity of an advanced numerical strategy, and the difficulty with writing a high-performance parallel implementation. Among related works, we can mention [26], [25], [24], [22], [33], [11], and [35]. In [26], a three-dimensional finite difference model with a parallel implementation is presented for solving the Bidomain equations with inhomogeneous and anisotropic conductivities. Four different parallelization schemes are examined in [25] for the simplified Monodomain model in two dimensions. A modular simulation system for the Bidomain equations is presented in [24] and speed-up results obtained on regular three-dimensional meshes are reported. Recently, a fully implicit parallel algorithm for the two-dimensional Bidomain equations is proposed in [22], adopting the advanced Newton-Krylov-Schwarz strategy but associated with a very simple cell model. Moreover, a computer three-dimensional heart model employing the Monodomain equation and a modified Luo-Rudy membrane model is reported in [33], where finite differences with explicit time stepping are used. In [11], both the monodomain and bidomain models are discussed, where the temporal discretization is of the semi-implicit type and the resulting bidomain stiffness matrix has a 2×2-block structure, in the same fashion as our numerical strategy to be presented in Section 3. However, the two PDEs are both of the reaction-diffusion type in the bidomain approach of [11], and one-level domain decomposition preconditioning techniques, which use a relatively simple subdomain solver, are reported for three-dimensional structured computational meshes. Finally, [35] reports some numerical experiences associated with a parallel multigrid preconditioner for a two-dimensional bidomain model based on rectangular meshes. The temporal discretization in [35] is done in an explicit fashion, such that the two PDEs are solved separately.

To the topic of parallel electro-cardiac simulations, the contributions from our earlier papers [19, 6, 7] and the present chapter are twofold. First, we have adopted realistic three-dimensional geometries and an implicit time-stepping scheme in parallel full-scale simulations, instead of simple rectangular domains. Our mathematical model (see Section 2) has also incorporated the torso domain, so that the ECG signal can be computed on the body surface, at the same time when the Bidomain equations are solved in the heart domain. Regarding our earlier papers [19] and [6], no parallel preconditioners were adopted in [19], whereas the numerical strategy used in [6] did not solve the two PDEs of the Bidomain model simultaneously. As a second contribution to parallel electro-cardiac simulations, we have devised a fast-convergent parallel block preconditioner, on the basis of an order-optimal serial block preconditioner that was derived in [32] and further analyzed in [21]. The parallel block preconditioner was first proposed in [7], and the present chapter contains, among other things, continued work on the parallelization. Using several performance enhancing techniques, which will be discussed in Section 5, our advanced parallel simulator is shown in Section 6 to obtain decent parallel performance even on a Linux cluster.

The focus of the present chapter is on a parallel full-scale electro-cardiac simulator, which employs anisotropic and inhomogeneous conductivity properties in realistic three-dimensional geometries modeling both the heart and the torso. Another advanced feature is that the two PDEs, which constitute the Bidomain system in the PDE part of the mathematical model, are solved together as a coupled 2×2 block system of linear equations during each time step. The 2×2 block linear system arises from a linearization based on an operator splitting technique, together with using an implicit time-stepping scheme associated with finite element discretizations; see Section 3. We will take the starting point in an order-optimal serial block preconditioner (see [32, 21]) for the 2×2 block system. The overall parallelization strategy is based on dividing a global solution domain into subdomains. The main challenge arises from devising and implementing a parallel substitute of the serial block preconditioner; see Section 2, which is essential for the PDE part to obtain a rapid convergence of a Krylov iterative linear solver. To achieve scalability with respect to the convergence speed, we build our parallel preconditioner on the basis of *additive Schwarz iterations*; see [9, 28, 37]. Such iterations use a "divide-and-conquer" strategy for partitioning the work load and thus are compatible with the overall subdomain-based parallel strategy. In fact, the following advanced numerical ingredients need to be incorporated into the parallel implementation:

1. a two-block diagonal system is chosen as the preconditioner and is solved during each preconditioning operation within a parallel conjugate gradient solver,
2. additive Schwarz iterations are used as the parallel solvers for each of the two diagonal blocks,
3. multigrid V-cycles are used as the subdomain solvers within each Schwarz iteration, and
4. global coarse grid corrections are also incorporated into the additive Schwarz iterations.

Using a combination of the above numerical techniques, we have been able to obtain an order-optimal parallel preconditioner for the 2×2 block linear system that arises from discretizing the Bidomain equations. More specifically, the number of parallel Krylov iterations needed for solving the 2×2 block system remains independent of both the number of degrees of freedom and the number of subdomains. Moreover, the total amount of work during each preconditioning operation remains approximately linearly proportional to the number of degrees of freedom.

As the foundation for the parallel implementation, we have used a specially designed mesh partitioning scheme, which partitions both the heart domain and the torso domain in a communication-efficient manner. Moreover, each subdomain is equipped with a hierarchy of adaptively refined subdomain meshes, on which multigrid V-cycles can be run. Due to the mathematical requirement of overlap by the additive Schwarz iterations, we have also taken care to avoid unnecessary duplicated local computations. Consequently, satisfactory speed-up results have been obtained for the resulting parallel electro-cardiac simulator; see Section 6.

The remainder of the chapter is organized as follows. First, Section 2 presents the mathematical model with an emphasis on the involved PDEs. Then, Section 3 explains the overall numerical strategy, which is semi-implicit and is based on an operator splitting technique. The specially designed block preconditioner, in both its serial and parallel versions, is also explained. Afterwards, Section 4 focuses on the parallelization of the serial numerical strategy, where implementing the parallel block preconditioner is the main theme. To obtain a satisfactory parallel performance, Section 5 discusses the issue of reducing the overhead in parallel simulations. Finally, Section 6 reports some detailed measurements of the full-scale parallel electro-cardiac simulator.

2 The Mathematical Model

The *Bidomain equations* constitute a well-established mathematical model for describing the electrical activities in the heart; see [34] and also the references given in [18]. Let H denote the domain of the heart. We consider the Bidomain equations of the following form (1)-(2), involving two primary unknown functions in H, i.e., the transmembrane potential v and the extracellular potential u_e.

$$\chi C_{\mathrm{m}} \frac{\partial v}{\partial t} + \chi I_{\mathrm{ion}}(v, \mathbf{s}) = \nabla \cdot (M_i \nabla v) + \nabla \cdot (M_i \nabla u_e) \qquad \text{in } H, \tag{1}$$

$$0 = \nabla \cdot (M_i \nabla v) + \nabla \cdot ((M_i + M_e) \nabla u_e) \qquad \text{in } H. \tag{2}$$

In (1), the nonlinear function $I_{\mathrm{ion}}(v, \mathbf{s})$ represents the ionic current, where $\mathbf{s}$ denotes a vector of state variables describing the state of the cell membrane. There exist many models of I_{ion}, which together with different ways of modeling the interaction between the state variables give rise to different cell models. Almost all of the cell models rely on solving a system of ODEs to update the $\mathbf{s}$ vector; see e.g. [20, 36]. More specifically, an ODE system of the form

$$\frac{\partial \mathbf{s}}{\partial t} = \mathbf{F}(v, \mathbf{s}, t) \tag{3}$$

models the electrical behavior of the cardiac cells, at every point inside H. The more sophisticated the cell model, the larger the number of ODEs are involved in (3). Moreover, χ in (1) is a surface-to-volume scaling factor, C_{m} denotes the capacitance of the membrane, and M_i denotes the intracellular conductivity tensor. Similarly, M_e denotes the extracellular conductivity tensor in (2). Due to the fibrous structure of the heart muscle, the conductivity

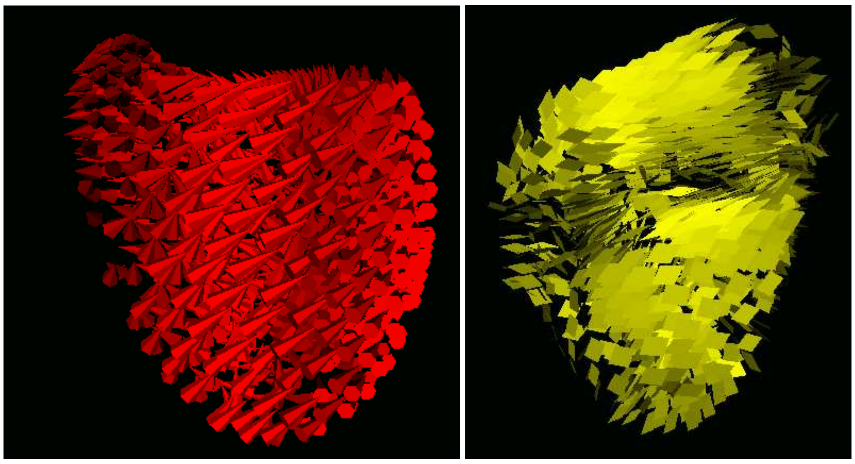

Fig. 2. The orientation of the muscle fibers (left) and sheet layers (right) in the heart.

tensors are anisotropic. More specifically, the formulas for M_i and M_e are as follows:

$$M_{i,e} = \sigma^t_{i,e} I + (\sigma^l_{i,e} - \sigma^t_{i,e})\mathbf{a}_l \mathbf{a}_l^T + (\sigma^n_{i,e} - \sigma^t_{i,e})\mathbf{a}_n \mathbf{a}_n^T, \tag{4}$$

where the constants $\sigma^t_{i,e}$, $\sigma^l_{i,e}$, and $\sigma^n_{i,e}$ describe the different conductivity properties of the heart muscle in three orthogonal directions, I is the identity matrix, while the vectors $\mathbf{a}_l$ and $\mathbf{a}_n$ model the orientation of the muscle fibers and sheet layers, varying throughout the heart. The superscripts l, t, n denote, respectively, the direction along the fibers, the direction normal to the fibers but in the plane of the sheets, and the direction normal to the sheets. Examples of the $\sigma^t_{i,e}, \sigma^l_{i,e}, \sigma^n_{i,e}$ values are given in (23)-(23) in Section 6. For a more detailed description, we refer to [17, 29]. Figure 2 shows an example of the orientation of the muscle fibers and sheet layers in the heart, of which the $\mathbf{a}_l$ and $\mathbf{a}_n$ vectors have already been used in the different simulations of [19, 6, 30, 7]. This set of heart muscle data is provided by the Biomedical Engineering Group at the University of Auckland; see [12]. Inhomogeneity of the conductivity properties will arise when electrical signals propagate from the heart into the torso, between organs in the torso, and also when dead or diseased tissues need to be modelled in the heart.

In order to compute the electrical potential on the body surface by the same simulation, we supplement the Bidomain equations (1)-(2) with the following elliptic PDE describing the propagation of the electrical signal in the torso T exterior to the heart:

$$\nabla \cdot (M_o \nabla u_o) = 0 \qquad \text{in } T, \tag{5}$$

where u_o is the electrical potential in the torso and M_o denotes the associated conductivity tensor. This supplementary PDE enables a direct comparison

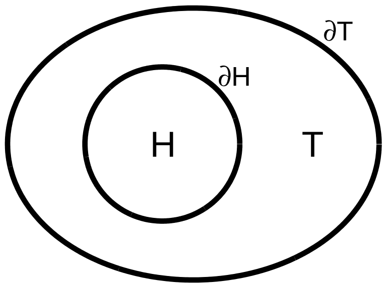

Fig. 3. A schematic 2D slice of the entire solution domain $\Omega = H \cup T$.

between ECG measurements and simulated results. We refer to [18] for a summary on this topic.

To summarize, our complete mathematical model consists of the ODE system (3) plus three PDEs (1)-(2), and (5), for which the combined solution domain $\Omega = H \cup T$ is depicted in Figure 3. As for the boundary conditions, we have

$$M_i \frac{\partial}{\partial n}(v + u_e) = 0, \quad u_e = u_o, \quad M_e \frac{\partial u_e}{\partial n} - M_o \frac{\partial u_o}{\partial n} = 0 \quad \text{on } \partial H, \tag{6}$$

$$M_o \frac{\partial u_o}{\partial n} = 0 \quad \text{on } \partial T, \tag{7}$$

where $\partial/\partial n$ denotes the derivative in the normal direction on the boundary. Although the boundary conditions are not sufficient for uniquely determining u_e and u_o, they are sufficient for practical applications which are interested in the variations within the u_e and u_o solutions, rather than their absolute values.

In the temporal direction, the mathematical model is to be solved within a time domain with known initial values for v and $\mathbf{s}$.

3 The Numerical Strategy

3.1 The Time-Stepping Scheme

The starting point of the whole numerical strategy is to use the technique of *operator splitting* (see e.g. [36]) to decouple the nonlinear PDE (1) into two parts:

$$\frac{\partial v}{\partial t} = -\frac{1}{C_\mathrm{m}} I_\mathrm{ion}(v, \mathbf{s}), \tag{8}$$

$$\chi C_\mathrm{m} \frac{\partial v}{\partial t} = \nabla \cdot (M_i \nabla v) + \nabla \cdot (M_i \nabla u_e), \tag{9}$$

i.e., an ODE and a linear PDE. The ODE (8) is then joined with a chosen ODE system of form (3) to establish the following new system of ODEs:

$$\begin{cases} \dfrac{\partial v}{\partial t} = -\dfrac{1}{C_{\rm m}} I_{\rm ion}(v, \mathbf{s}), \\ \dfrac{\partial \mathbf{s}}{\partial t} = \mathbf{F}(v, \mathbf{s}, t), \end{cases} \tag{10}$$

which needs to be solved during each time step. We remark that the choice of $I_{\rm ion}$ and $\mathbf{F}$ in the above ODE system determines the so-called cell model, for which a well-known example is the Winslow model [36].

In the temporal direction, the simulation time domain is divided into discrete time levels:

$$0 = t_0 < t_1 < t_2 \cdots,$$

where at each time level t_l, $l \geq 1$, the solutions from the previous time level, $v^{l-1}, u_e^{l-1}, u_o^{l-1}, \mathbf{s}^{l-1}$, are used as the starting values. Using a θ-rule, where $0 \leq \theta \leq 1$, we can construct a flexible time-stepping scheme whose work per time step consists of solving an ODE system twice, separated by the solution of a system of three PDEs. More specifically, the computational work for the time step $t_{l-1} \rightarrow t_l$ consists of the following sub-steps:

1. At every mesh point in H, an ODE system of form (10) is solved for $t \in (t_{l-1}, t_{l-1} + \theta(t_l - t_{l-1})]$, using the initial values v^{l-1} and $\mathbf{s}^{l-1}$. The solution results are an intermediate transmembrane potential solution $\tilde{v}^{l-1}$ and an updated $\tilde{\mathbf{s}}^{l-1}$ vector.
2. The three PDEs (9), (2), and (5) are solved simultaneously, where we remark that (9) is the remaining part of (1) after operator splitting. The temporal discretization, which uses a θ-rule, is of the following form:

$$\begin{aligned} \chi C_{\rm m} \frac{\hat{v}^l - \tilde{v}^{l-1}}{\Delta t} &= (1-\theta)\left(\nabla \cdot (M_i \nabla \tilde{v}^{l-1}) + \nabla \cdot (M_i \nabla u_e^{l-1})\right) \\ &\quad + \theta \left(\nabla \cdot (M_i \nabla \hat{v}^l) + \nabla \cdot (M_i \nabla u_e^l)\right), \quad (11) \\ 0 &= \nabla \cdot (M_i \nabla \hat{v}^l) + \nabla \cdot ((M_i + M_e) \nabla u_e^l), \quad (12) \\ 0 &= \nabla \cdot (M_o \nabla u_o^l). \quad (13) \end{aligned}$$

We remark that $\hat{v}^l, u_e^l, u_o^l$ are the unknowns to be found. If we combine the unknown values of u_e^l and u_o^l into one vector $\mathbf{u}^l$, the spatial discretization (using e.g. finite elements) will give rise to a 2×2 block linear system:

$$\begin{bmatrix} \chi C_{\rm m} \mathbf{I} + \theta \Delta t \mathbf{A}_{\rm v} & \theta \Delta t \hat{\mathbf{A}}_{\rm v} \\ \theta \Delta t \hat{\mathbf{A}}_{\rm v}^T & \theta \Delta t \mathbf{A}_{\rm u} \end{bmatrix} \begin{bmatrix} \hat{\mathbf{v}}^l \\ \mathbf{u}^l \end{bmatrix} \equiv \mathcal{A} \begin{bmatrix} \hat{\mathbf{v}}^l \\ \mathbf{u}^l \end{bmatrix} = \begin{bmatrix} \mathbf{b}^l \\ \mathbf{0} \end{bmatrix}. \tag{14}$$

In the above block linear system, $\mathbf{I}$ and $\mathbf{A}_{\rm v}$ represent the mass matrix and the stiffness matrix associated with M_i inside H, respectively. The $\mathbf{b}^l$ vector is computed on the basis of $\tilde{v}^{l-1}$ and u_e^{l-1}. The sparse matrix $\mathbf{A}_{\rm u}$ arises from discretizing (2) and (5) together. That is, we discretize a combined elliptic equation:

$$\nabla \cdot (M \nabla u) = 0 \quad \text{in } \Omega, \tag{15}$$

using $M = M_i + M_e$ in H and $M = M_o$ in T; see [32] for more details. The number of degrees of freedom in $\mathbf{u}^l$ equals the number of mesh points covering the combined domain $\Omega = H \cup T$. The $\hat{\mathbf{A}}_v$ matrix is the same as $\mathbf{A}_\mathrm{v}$, except that $\hat{\mathbf{A}}_v$ is padded with some additional zero columns to allow a multiplication of form $\hat{\mathbf{A}}_v\mathbf{u}$, whereas the $\hat{\mathbf{A}}_v^T$ matrix is the transpose of $\hat{\mathbf{A}}_v$.

3. At every mesh point in H, the ODE system (10) is solved again for $t \in (t_{l-1} + \theta(t_l - t_{l-1}), t_l]$, using the initial values $\hat{v}^l$ and $\tilde{\mathbf{s}}^{l-1}$. The computational results are stored in v^l and $\mathbf{s}^l$.

We refer to [32] for a detailed explanation of the above time-stepping scheme, and remark that using $\theta = 1/2$ in (11) gives rise to a second-order accurate temporal discretization. To handle the potentially stiff ODE system (10), due to steep propagation fronts, we adopt a third-order Runge-Kutta type scheme proposed in [31]. Although a second-order ODE solver is sufficient for ensuring the order of accuracy in the time direction, our experiences have indicated that the adopted third-order ODE solver is better at treating the situations of stiffness, so that fewer intermediate ODE steps are needed between two time levels. That is, the overall time spent by a third-order ODE solver is not necessarily larger than that of a lower order ODE solver.

3.2 A Serial Block Preconditioner

By extending the proof given in [23] to also include the torso domain, we can deduce that the 2×2 block matrix $\mathcal{A}$ defined in (14) is *symmetric and positive semi-definite*. We remark that this property arises from the symmetry and positive definiteness of the conductivity tensors M_i, M_e, and M_o (consequently are the $\mathbf{A}_\mathrm{v}$ and $\mathbf{A}_\mathrm{u}$ matrices symmetric and positive semi-definite); see [23]. The existence of zero eigenvalues for $\mathcal{A}$ is due to the boundary conditions (6)-(7). Our earlier experiences have suggested that the conjugate gradient (CG) method is an appropriate choice for solving (14); see [32].

The next question is how fast can the CG method achieve convergence for (14)? To answer this question, we have conducted three experiments where we study the number of CG iterations, denoted by I_CG, which is needed to reduce the residual vector associated with the 2×2 block system (14) by a factor of 10^4 in the L_2-norm, compared with its initial value before the first CG iteration. We remark that a typical value of $\Delta t = 0.125$ms is chosen in building $\mathcal{A}$, and our experiences suggest that the size of Δt within the range of $[0.1, 0.5]$ms does not have an immediate effect on the number of CG iterations. In Table 1, N_H denotes the number of unknowns in the $\mathbf{v}$ vector, i.e., the number of mesh points in the three-dimensional heart domain, whereas N_Ω denotes the number of unknowns in the $\mathbf{u}$ vector, i.e., the number of mesh points in the entire torso domain including the heart. The different numbers of N_H and N_Ω are obtained from an adaptive mesh refinement process, which increases the mesh resolution mainly inside H and keeps the mesh resolution

Table 1. An example on the number of CG iterations (without preconditioner) needed for solving the 2×2 block system (14) of different sizes.

$N_H + N_\Omega$	I_{CG}
39,589	897
302,166	1999
1,552,283	4087

inside T mostly unchanged. (We note that all the numerical experiments reported in the present chapter are associated with realistic three-dimensional geometries.) It can be deduced that without preconditioning the number of CG iterations approximately doubles when the spacing between mesh points is halved. The convergence speed is determined by the elliptic operators. Consequently, the work amount of each CG iteration grows much faster than the number of degrees of freedom, when no preconditioner is used. Although the values of I_{CG} in Table 1 are associated with one particular time step, numerical experiments have suggested that I_{CG} remains roughly constant throughout an entire elector-cardiac simulation, i.e., I_{CG} is not sensitive to the changing right-hand side vector in (14).

To overcome the huge numbers of CG iterations reported in Table 1, a preconditioner (see e.g. [2]) needs to be used inside the CG iterations. The standard choices of preconditioning (such as SSOR and RILU) all have the weakness that the number of CG iterations grows considerably with respect to the number of degrees of freedom (although the growth is slower than that reported in Table 1). We would thus like to have an *order-optimal* preconditioner, which means that the number of CG iterations remains constant independent of the number of degrees of freedom. Moreover, the work executed in each operation of such an order-optimal preconditioner should be linearly proportional to the number of degrees of freedom.

A so-called block preconditioner based on multigrid algorithms was first proposed in [32]. More specifically, the following diagonal block matrix

$$\mathcal{D} = \begin{bmatrix} \chi C_{\mathrm{m}} \mathbf{I} + \theta \Delta t \mathbf{A}_{\mathrm{v}} & \mathbf{0} \\ \mathbf{0} & \theta \Delta t \mathbf{A}_{\mathrm{u}} \end{bmatrix} \tag{16}$$

is used as a suitable preconditioner for (14). In other words, the original 2×2 block system (14) is replaced with

$$\mathcal{D}^{-1} \mathcal{A} \begin{bmatrix} \hat{\mathbf{v}}^l \\ \mathbf{u}^l \end{bmatrix} = \mathcal{D}^{-1} \begin{bmatrix} \mathbf{b}^l \\ \mathbf{0} \end{bmatrix}. \tag{17}$$

This means that during each preconditioning operation, the two diagonal blocks in $\mathcal{D}$ need to be solved (approximately). It has recently been shown in [21] that the number of CG iterations, when using the block preconditioner $\mathcal{D}$, will remain independent of the number of degrees of freedom. The rapid convergence of this block preconditioner is due to the fact that $\mathcal{D}$ is

Table 2. An example on the number of CG iterations (using the serial multigrid-block preconditioner) needed for solving the 2×2 block system (14).

$N_H + N_\Omega$	I_{CG}
302,166	15
1,552,283	16

spectrally equivalent to the 2×2 block matrix $\mathcal{A}$ in (14). To make $\mathcal{D}$ an order-optimal serial preconditioner, we have chosen to apply one multigrid V-cycle (see [13, 1]) as the approximate solver for both the diagonal blocks in $\mathcal{D}$. This is because multigrid V-cycles are powerful linear solvers and the work amount of one multigrid V-cycle is linearly proportional to the number of degrees of freedom.

To demonstrate the actual effect of the block preconditioner $\mathcal{D}$ defined in (16) using multigrid V-cycles, we list in Table 2 the number of CG iterations when the block preconditioner is in use. In comparison with Table 1, we can see that I_{CG} is dramatically reduced and stays independent of the number of degrees of freedom. Here, we remark that $N_H + N_\Omega = 39,589$ in Table 1 is related to the coarsest possible meshes we have for modeling our realistic heart and torso geometries (see Figure 1). The measurement of I_{CG} associated with $N_H + N_\Omega = 39,589$ is absent in Table 2 because no coarser meshes can be found to form a mesh hierarchy needed by the multigrid V-cycles.

3.3 A Parallel Block Preconditioner

The objective of finding a parallel version of the block preconditioner $\mathcal{D}$ defined in (16) is accompanied with the request of maintaining the rapid convergence speed and the scalability in work amount. To achieve this, we replace the multigrid V-cycles, which are used in the serial preconditioner for approximately solving each of the diagonal blocks in $\mathcal{D}$, with *additive Schwarz iterations* (see [9, 28]). This is mainly motivated by the inherent parallelism of these domain decomposition algorithms. Moreover, the simple algorithmic structure of the additive Schwarz iterations (e.g., no special interface solvers are needed) suits very well for a parallel implementation. In addition, additive Schwarz iterations are also known to have very good convergence behavior, similar to the multigrid algorithms.

Roughly speaking, the starting point of any additive Schwarz iteration is that a global solution domain Ω is divided into a set of *overlapping* subdomains $\{\Omega_s\}$. Each subdomain becomes an independent working unit, which mostly concentrates on local discretizations within Ω_s and solving local linear systems. In addition, the subdomains frequently collaborate with each other, in a form that neighboring subdomains exchange local solutions within overlapping zones. A loose synchronization of the work progress on the subdomains also has to be enforced.

The Mathematical Framework of Additive Schwarz Iterations

The mathematics behind the additive Schwarz iterations can be understood as follows. Suppose we want to solve a global linear system

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}, \tag{18}$$

which arises from discretizing a PDE in a global domain Ω. Given a set of Psubdomains $\{\Omega_s\}$, $1 \leq s \leq P$, such that $\Omega = \cup\Omega_s$ and there is a certain amount of overlap between neighboring subdomains, we locally discretize the PDE in every subdomain Ω_s. The result of the local discretization is

$$\boldsymbol{A}_s\boldsymbol{x}_s = \boldsymbol{b}_s(\boldsymbol{x}|_{\partial\Omega_s\backslash\partial\Omega}). \tag{19}$$

The above linear system namely arises from restricting the discretization of the target PDE within Ω_s. The only special treatment happens on the so-called *internal boundary* $\partial\Omega_s\backslash\partial\Omega$, i.e., the part of $\partial\Omega_s$ that does not coincide with the physical boundary $\partial\Omega$ of the global domain. We remark that a requirement of the overlapping zones says that any point lying on the internal boundary of a subdomain must also be an interior point in at least one of the neighboring subdomains. On the internal boundary, artificial Dirichlet conditions are repeatedly updated using new values of the subdomain solutions computed in the neighboring subdomains. The involvement of the artificial Dirichlet conditions is indicated by the notation $\boldsymbol{b}_s(\boldsymbol{x}|_{\partial\Omega_s\backslash\partial\Omega})$ in (19). On the remaining part of $\partial\Omega_s$, the original boundary conditions of the target PDE are valid as before.

For the artificial Dirichlet conditions to converge toward the correct values on the internal boundary, iterations need to be carried out. That is, we generate on each subdomain a series of approximate solutions $\boldsymbol{x}_s^0, \boldsymbol{x}_s^1, \boldsymbol{x}_s^2 \ldots$, which will hopefully converge toward the correct solution $\boldsymbol{x}_s = \boldsymbol{x}|_{\Omega_s}$. The kth additive Schwarz iteration is thus defined as

$$\boldsymbol{x}_s^k \;=\; \boldsymbol{A}_s^{-1}\boldsymbol{b}_s(\boldsymbol{x}^{k-1}|_{\partial\Omega_s\backslash\partial\Omega}), \quad \boldsymbol{x}^k = \text{composition of all } \boldsymbol{x}_s^k. \tag{20}$$

The symbol $\boldsymbol{A}_s^{-1}$ in (20) means an inverse of the subdomain matrix $\boldsymbol{A}_s$, but an approximate subdomain solver that is sufficiently close to $\boldsymbol{A}_s^{-1}$ is also allowed. The right-hand side vector $\boldsymbol{b}_s$ needs to be updated with artificial Dirichlet conditions on the internal boundary, using solution of the previous Schwarz iteration provided by the neighboring subdomains.

Parallel Computing with Additive Schwarz Iterations

We note that the subdomain local solves in (20) can be carried out independently in each additive Schwarz iteration. This immediately gives rise to the possibility of parallel computing. At the end of the kth additive Schwarz iteration, the (logically existing) global approximate solution $\boldsymbol{x}^k$ is composed

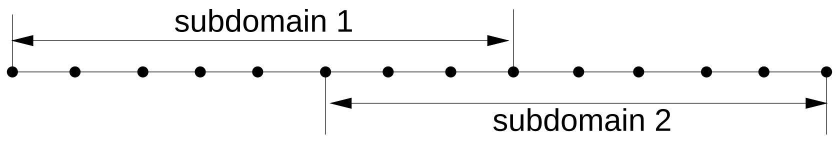

Fig. 4. A simple example of two overlapping subdomains.

on the basis of the subdomain approximate solutions $\{\boldsymbol{x}_s^k\}$. In particular, the following rule for composing a global solution, using the principle of *partition of unity*, should be used:

- An overlapping point refers to a point that lies inside a zone of overlap, i.e., the point belongs to at least two subdomains.
- For every non-overlapping point, i.e., a point that belongs to only one subdomain, the global solution attains the same value as that inside the host subdomain.
- For every overlapping point, let us denote by n_{total} the total number of host subdomains that own this point. Let also n_{interior} denote the number of subdomains, among those n_{total} host subdomains, which do not have the point lying on their internal boundaries. (The setup of the overlapping subdomains ensures $n_{\text{interior}} \geq 1$.) Then, the average of the n_{interior} local values becomes the global solution on the point. The other $n_{\text{total}} - n_{\text{interior}}$ local values are not used, because the point lies on the internal boundary there. (Take Figure 4 for instance, where there are four overlapping points shared between the two subdomains. For the two "middle" overlapping points we have $n_{\text{interior}} = n_{\text{total}} = 2$, whereas for the two "outer" overlapping points we have $n_{\text{interior}} = 1$ and $n_{\text{total}} = 2$. That is, on each of the two "middle" overlapping points, the two values from the two subdomains should be averaged, whereas on each of the two "outer" overlapping points, the value from the neighboring subdomain should replace the value on the host subdomain.) Finally, the obtained global solution is enforced in each of the n_{total} host subdomains. For the $n_{\text{total}} - n_{\text{interior}}$ host subdomains, which have the point lying on their internal boundary, the obtained global solution will be used as the artificial Dirichlet condition during the next Schwarz iteration.

To compose the global solution and update the artificial Dirichlet conditions, as described by the above rule, we need to carry out a procedure of communication among the neighboring subdomains at the end of each additive Schwarz iteration. During this procedure of communication, each pair of neighboring subdomains exchanges between each other an array of values that are associated with their shared overlapping points. It is clear that if each subdomain solution $\boldsymbol{x}_s^k$ converges toward the correct solution $\boldsymbol{x}|_{\Omega_s}$, the difference between the subdomain solutions in an overlapping zone will eventually disappear.

Mathematically, a subdomain matrix $\boldsymbol{A}_s$ in (19) should arise from first building the global matrix $\boldsymbol{A}$ in (18) and then cutting out the portion of $\boldsymbol{A}$ that corresponds to the mesh points lying in Ω_s. However, this approach requires unnecessary construction and storage of global matrices, which is not a desired situation during parallel computations. We just make the point that the global matrix $\boldsymbol{A}$ can be conceptually represented by the collection of subdomain matrices $\boldsymbol{A}_s$.

A Layered Design of the Parallel Block Preconditioner

To devise a parallel version of the block preconditioner $\mathcal{D}$ defined in (16), we suggest a layered design. First, two separate additive Schwarz iterations (see e.g. [5]) approximately solve the two diagonal blocks in $\mathcal{D}$, i.e., $\chi C_\mathrm{m}\mathbf{I}+\theta\Delta t\mathbf{A}_\mathrm{v}$ and $\theta\Delta t\mathbf{A}_\mathrm{u}$, respectively. Let us demonstrate this for the second diagonal block. For this purpose, one additive Schwarz iteration for approximating the inverse of $\mathbf{A}_\mathrm{u}$ can be expressed as

$$\mathbf{A}_\mathrm{u}^{-1} \approx \sum_{s=0}^{P} \mathbf{A}_{\mathrm{u},s}^{-1}, \tag{21}$$

where it is assumed that the body domain Ω is partitioned into P overlapping subdomains Ω_s, $1 \leq s \leq P$. Thus, $\mathbf{A}_{\mathrm{u},s}$ denotes a subdomain matrix that arises from a discretization restricted to Ω_s. Note that $\mathbf{A}_{\mathrm{u},0}$ in (21) is associated with a discretization on a very coarse global grid. The use of $\mathbf{A}_{\mathrm{u},0}^{-1}$, also called *coarse grid correction*, is to ensure convergence independent of the number of subdomains P; see e.g. [28]. The implementation issues for this topic will be discussed in Section 4.3.

Second, as an approximate subdomain solver, we use multigrid V-cycles. This is because the complexity of such cycles is linearly proportional to the number of subdomain unknowns. The construction of a required hierarchy of subdomain meshes is described in Section 4.2.

In summary, the combination of additive Schwarz iterations on the "global layer" and multigrid V-cycles on the "subdomain layer" constitutes the parallel version of the block preconditioner $\mathcal{D}$ defined in (16). Scalability with respect to the number of unknowns is due to the spectral equivalence between the inverse of the two diagonal blocks in $\mathcal{D}$ with the two separate additive Schwarz iterations, such as (21) corresponds to the second diagonal block in $\mathcal{D}$. We remark that corresponding scalability in the serial case has been proved in [21]. Moreover, multigrid V-cycles are used as the subdomain solvers in the additive Schwarz iterations. Convergence independent of the number of subdomains is due to using the coarse grid correction; see e.g. [28].

4 A Parallel Electro-Cardiac Simulator

Incorporating parallelism into the preceding numerical strategy for electro-cardiac simulations requires parallelization of *each* of the three sub-steps in the time-stepping scheme from Section 3.1. The first and third sub-steps are readily parallelizable, because the solution of the ODE system (10) at any two mesh points in H can be carried out completely independent of each other. For the second sub-step, preconditioned CG iterations can be parallelized using a subdomain-based approach with a distributed data structure, which will be explained in the following text.

4.1 Parallel Computing Based on Subdomains

Let P denote the number of processors; we adopt the approach of explicit domain partitioning that divides the entire computational work among the processors. Recall that the combined global domain Ω consists of the heart domain H and the torso domain T, so we partition both H and T into P pieces; see Figure 5. Processor s is thus responsible for the composite subdomain $\Omega_s = H_s \cup T_s$. In addition, we also introduce a certain amount of overlap between the subdomains to enable the additive Schwarz iterations useful in the parallel preconditioner. More details on the issue of domain partitioning will be presented in Section 4.2.

For the parallelization of the first and third sub-steps of the time-stepping scheme, i.e., solving the ODE system (10), processor s only needs to consider the mesh points inside the heart subdomain H_s. In principle, no communication is needed between the processors. However, to reduce the unnecessary duplicated local computations due to overlap between the subdomains, some additional communication may help to reduce the local computation volume; see Section 5.

To achieve parallel CG iterations using the block preconditioner, both the involved linear algebra operations and the block preconditioner need to be executed in parallel. A distributed data structure suits this purpose very well. We recall that the global domains H and Ω are partitioned into a set of subdomains $\{H_s\}_{s=1}^{P}$ and $\{\Omega_s\}_{s=1}^{P}$. The global matrices (such as $\mathbf{A}_\mathrm{v}$ and $\mathbf{A}_\mathrm{u}$) and global vectors can thus be represented *collectively* by the subdomain matrices (such as $\mathbf{A}_{\mathrm{v},s}$ and $\mathbf{A}_{\mathrm{u},s}$) and subdomain vectors that are *distributed* on the different processors. There is no need to physically construct the global matrices and vectors, because all the global linear algebra operations involved in a Krylov subspace method (such as the CG method) can be parallelized by doing subdomain linear algebra operations with additional communication between neighboring subdomains. In such a setup, each subdomain becomes an independent working unit, which mostly concentrates on local discretizations within H_s or Ω_s and solving local linear systems. In addition, the subdomains frequently collaborate with each other, in a form that neighboring subdomains exchange local solutions within overlapping zones. A loose synchronization of the work progress on the subdomains also has to be enforced.

During parallel computations for the PDE part, i.e., the second sub-step of the time-stepping scheme, the work on subdomain s consists of local operations that are restricted to H_s and T_s. That is, local finite element discretizations are carried out independently on each processor. No communication between the processors is needed for this task of distributed discretizations. Afterwards, during the parallel CG iterations for solving the global 2×2 block linear system (14), which is distributed as a set of subdomain 2×2 block systems, subdomain local operations need to be interleaved with inter-processor communication.

Such a subdomain-based approach also suits very well the parallel version of the block preconditioner $\mathcal{D}$ presented in Section 3.3. Subdomain multigrid V-cycles need only the local matrices and vectors to find, e.g., the approximate inverse of $\mathbf{A}_{\mathrm{u},s}$, which is needed in the additive Schwarz iterations. Besides, the distributed data structure is compatible with the communication between neighboring subdomains when all the processors have finished the subdomain multigrid V-cycles. The rules for the communication are described earlier in Section 3.3.

4.2 A Special Strategy for Mesh Partitioning

A special feature of our numerical strategy is that both the ODE system (10) and the linear parabolic PDE (9) use H as the solution domain, whereas the composite elliptic PDE (15) uses $\Omega = H \cup T$ as the solution domain. Therefore, we need to have $H_s \subset \Omega_s$ on each subdomain in order to form a distributed set of subdomain 2×2 block matrices $\mathcal{A}_s$ $(1 \leq s \leq P)$ for running the parallel CG iterations, where

$$\mathcal{A}_s = \begin{bmatrix} \chi C_{\mathrm{m}} \mathbf{I}_s + \theta \Delta t \mathbf{A}_{\mathrm{v},s} & \theta \Delta t \hat{\mathbf{A}}_{\mathrm{v},s} \\ \theta \Delta t \hat{\mathbf{A}}_{\mathrm{v},s}^T & \theta \Delta t \mathbf{A}_{\mathrm{u},s} \end{bmatrix}. \tag{22}$$

In addition, to apply multigrid V-cycles as the subdomain solvers in each additive Schwarz iteration, we also need to have a hierarchy of subdomain meshes for H_s and a hierarchy of subdomain meshes for Ω_s, respectively. To satisfy these two requirements, we use the following scheme for partitioning the global domains H and Ω, while generating desired subdomain mesh hierarchies:

1. For the global domain Ω, we start with a global coarse mesh $\mathcal{G}_{\Omega,0}$. This mesh can be used in connection with coarse grid corrections inside the additive Schwarz iterations; see Section 4.3.
2. Then, if necessary, we recursively perform an adaptive mesh refinement $J_g \geq 0$ times, such that we obtain a hierarchy of global meshes: $\mathcal{G}_{\Omega,0}$, $\mathcal{G}_{\Omega,1}, \ldots, \mathcal{G}_{\Omega,J_g}$.

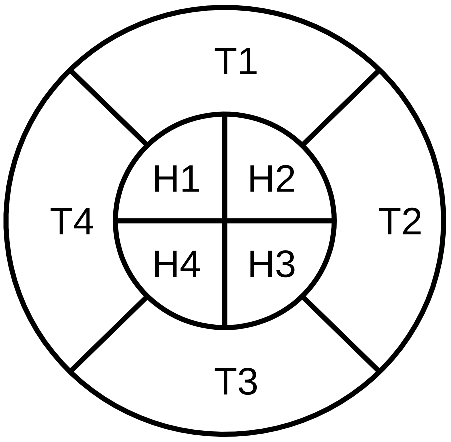

Fig. 5. A simplified diagram showing the strategy of domain partitioning for producing subdomains H_s and Ω_s. Here are four heart subdomains and four torso subdomains. Each subdomain Ω_s is composed of H_s and T_s.

3. The so-far finest global mesh $\mathcal{G}_{\Omega,J_g}$ is then partitioned to give rise to a set of overlapping subdomain meshes: $\{\mathcal{G}_{\Omega_s,0}\}_{s=1}^{P}$. This overlapping mesh partitioning is achieved by first individually carrying out non-overlapping partitioning of the global meshes $\mathcal{G}_{H,J_g}$ and $\mathcal{G}_{T,J_g}$. (Note that $\mathcal{G}_{T,J_g}$ may contain a considerable number of mesh points, necessary for describing the fine-scale changes of the conductivity tensor M_o due to, e.g., lungs and bones.) Thereafter, a certain amount of overlap is added to the subdomain meshes to give rise to the overlapping subdomain meshes $\{\mathcal{G}_{\Omega_s,0}\}$, where each $\mathcal{G}_{\Omega_s,0}$ contains a subdomain heart mesh $\mathcal{G}_{H_s,0}$. Note that the second subscript '0' in the notation $\mathcal{G}_{\Omega_s,0}$ indicates that $\mathcal{G}_{\Omega_s,0}$ is the coarsest subdomain mesh on subdomain s.
4. Afterwards, we recursively perform an adaptive mesh refinement J_s times on each subdomain independently. The final result is that, on subdomain number s, we obtain a subdomain mesh hierarchy: $\mathcal{G}_{\Omega_s,0}, \mathcal{G}_{\Omega_s,1}, \ldots, \mathcal{G}_{\Omega_s,J_s}$, which can be used by the subdomain multigrid V-cycles for solving $\mathbf{A}_{\mathrm{u},s}$. In addition, another subdomain mesh hierarchy is also created for the subdomain H_s, i.e., $\mathcal{G}_{H_s,0}, \mathcal{G}_{H_s,1}, \ldots, \mathcal{G}_{H_s,J_s}$.

The basic idea of domain partitioning in the above partitioning scheme can be depicted by a simplified diagram in Figure 5, where we have four heart subdomains and four torso subdomains. Each body subdomain Ω_s is composed of H_s and T_s. A realistic example (which was first reported in connection with [19]) is shown in Figure 6.

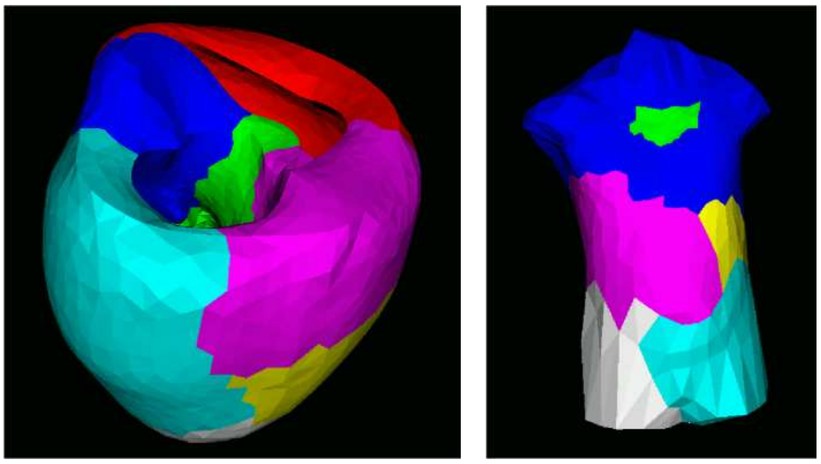
Fig. 6. An example of partitioning an unstructured heart mesh (left) and an unstructured torso mesh (right).

Remarks.

Balancing the work load on the subdomains depends on an even distribution of the sizes of the subdomain meshes $\mathcal{G}_{\Omega_s,j}$ and $\mathcal{G}_{H_s,j}$ on all the subdomain mesh levels, $0 \leq j \leq J_s$. This can be roughly achieved by first finding balanced overlapping subdomain meshes $\mathcal{G}_{H_s,0}$ and $\mathcal{G}_{T_s,0}$, for $1 \leq s \leq P$. (Note that $\mathcal{G}_{\Omega_s,0} = \mathcal{G}_{H_s,0} \cup \mathcal{G}_{T_s,0}$.) Then, the adaptive mesh refinement on each subdomain has to take care to maintain the load balance, while ensuring that elements in the overlapping regions are refined in exactly the same way between the neighboring subdomains. We remark that the subdomain meshes at the finest level, e.g., $\{\mathcal{G}_{\Omega_s,J_s}\}_{s=1}^{P}$, constitute the finest *virtual* global mesh. In practice, the subdomain meshes normally have a certain degree of load imbalance, which is one of the main obstacles for achieving perfect speedup results. The need for a relatively fine global coarse mesh $\mathcal{G}_{\Omega,J_g}$ (or $\mathcal{G}_{H,J_g}$) may arise when the coarsest global mesh $\mathcal{G}_{\Omega,0}$ (or $\mathcal{G}_{H,0}$) is too coarse to limit the amount of overlap between the subdomains during the overlapping mesh partitioning. More details can be found in [8].

4.3 Coarse Grid Corrections

We recall from Section 3.3 that $\mathbf{A}_{\mathrm{u},0}^{-1}$ is included in the formula of the additive Schwarz iteration (21), which approximately inverts $\mathbf{A}_{\mathrm{u}}$. The action of $\mathbf{A}_{\mathrm{u},0}^{-1}$, which is denoted a coarse grid correction, refers to solving a global problem associated with the coarsest global mesh $\mathcal{G}_{\Omega,0}$. For the block matrix $\mathbf{A}_{\mathrm{u}}$, which arises from discretizing the combined elliptic equation (15), the coarse grid correction is particularly important for obtaining constant convergence speed, independently of the number of subdomains P. Otherwise, without the coarse

Table 3. An example of the effect of coarse grid corrections (CGCs), associated with the parallel block preconditioner, where I_{CG} denotes the number of CG iterations needed for solving the 2×2 block system (14).

		Without CGCs	With CGCs
$N_H + N_\Omega$	P	I_{CG}	I_{CG}
302,166	2	31	7
	4	41	7
	8	52	7
	16	79	8
1,552,283	2	32	7
	4	44	7
	8	63	8
	16	113	8

grid correction, the effect of the parallel block preconditioner will deteriorate when P becomes large. The effect of coarse grid corrections are well illustrated in Table 3.

In the case where the coarsest global mesh $\mathcal{G}_{\Omega,0}$ has a small number of mesh points, it suffices for every subdomain to solve the same global coarse grid problem using a serial algorithm. Compared with using a parallel coarse grid solver, this serial approach requires fewer inter-processor communications at the cost of a small increase of subdomain computations. However, if $\mathcal{G}_{\Omega,0}$ has quite many mesh points, the serial approach will become too costly. A parallel coarse grid solver must therefore be adopted. In such a case where the $\mathcal{G}_{\Omega,0}$ mesh is quite fine, we assume that there is no need to refine it before carrying out overlapping mesh partitioning, i.e., $J_g = 0$. Therefore, the subdomain coarsest meshes $\mathcal{G}_{\Omega_s,0}$ constitute a decomposition of $\mathcal{G}_{\Omega,0}$, which can be used to build a distributed data structure on the coarsest global mesh level and run the coarse grid solver in parallel.

5 Some Techniques for Overhead Reduction

Recall that the parallel block preconditioner from Section 3.3 uses two separate additive Schwarz iterations as approximate solvers for the two diagonal block matrices of $\mathcal{D}$ as defined in (16). The mathematical theory of the additive Schwarz method requires overlapping subdomains for ensuring the convergence; see [9, 28, 37]. Therefore, a number of elements and mesh points are shared between each pair of neighboring subdomains. We also remark that sufficient overlap between neighboring subdomains is important for obtaining a rapid convergence.

However, the mathematically required overlap between the subdomains will give rise to some duplicated computations in both the ODE part and the PDE part of the parallel electro-cardiac simulator. Regarding the ODE part, in order to avoid communication in the embarrassingly parallel ODE solver, neighboring subdomains must duplicate computations on the mesh points lying inside the overlap. As the ODE solver may be quite computationally intensive, this duplication gives rise to considerable overhead, which deteriorates the speedup. Similarly, duplication also arises in the PDE part, associated with the overlap between the subdomains. Such duplicated PDE computations are present on each shared mesh point inside the overlap, when the matrix-vector product of the global CG method is distributed as a set of subdomain linear algebra operations. However, we stress that there is no duplicated computation during the parallel preconditioning operations, because neighboring subdomains are meant to compute different values for the shared mesh points.

To remove the overhead due to duplicated computations described above, we need to do a *disjoint* re-distribution of all the mesh points. Such a disjoint re-distribution should be done on the basis of the existing overlapping subdomain meshes. For this purpose, let us denote by N_s the total number of mesh points in subdomain s. Since there is overlap between the subdomains, we have (for $P > 1$)

$$N < \sum_{s=1}^{P} N_s,$$

where N denotes the total number of points in a global mesh. Moreover, let us denote by N_s^O and N_s^I (where $N_s^O + N_s^I = N_s$) the number of overlapping points and the number of interior (non-overlapping) points in subdomain s, respectively. The essence of a disjoint re-distribution is to divide the N_s^O overlapping points on subdomain s into two parts: $N_s^O = N_s^{O_c} + N_s^{O_n}$, where $N_s^{O_c}$ is the number of overlapping points that participate in all the computations, whereas the remaining $N_s^{O_n}$ *non-computational* overlapping points do not participate in either the ODE solver or the global-level CG operations in the PDE solver. The objective of the disjoint re-distribution is to achieve

$$N = \sum_{s=1}^{P} \left(N_s^{O_c} + N_s^I \right),$$

while approximately maintaining a constant value of $N_s^{O_c} + N_s^I$ independent of s.

The removal of the duplicated computations, however, comes at the cost of an increased volume of communication. This is because the values that are needed on the $N_s^{O_n}$ non-computational overlapping points need to be provided by the neighboring subdomains, through message passing. Normally, the increase of the communication overhead can be justified by the removal of the duplicated computations.

Devising a high-quality and efficient re-distribution scheme is not trivial. Research is currently under way to find an optimal scheme which is also executable in parallel, involving collaboration between all the subdomains. For

Table 4. An example of applying a disjoint re-distribution to the H domain mesh points, where $N_H = 632,432$ and WT_{ODE} denotes the time consumptions by the ODE solver for one time step on a Linux cluster; see Section 6 for more details.

	Before re-distribution		After re-distribution	
P	$\max_s N_{H_s}$	WT_{ODE}	$\max_s(N^I_{H_s} + N^{O_c}_{H_s})$	WT_{ODE}
2	405,893	120.75	316201	117.40
4	225,296	73.48	162,823	65.31
8	137,446	51.82	89,025	37.20
16	71,643	29.48	49,057	23.08

Table 5. An example of applying a disjoint re-distribution to the Ω domain mesh points, where $N_\Omega = 919,851$ and WT_{CG} denotes the time consumptions by the parallel CG solver for one time step on a Linux cluster; see Section 6 for more details.

	Before re-distribution		After re-distribution	
P	$\max_s N_{\Omega_s}$	WT_{CG}	$\max_s(N^I_{\Omega_s} + N^{O_c}_{\Omega_s})$	WT_{CG}
2	600,773	44.30	476,382	42.63
4	332,163	27.16	248,841	26.42
8	200,987	21.57	123,662	20.86
16	110,146	18.40	70,164	17.21

the time being, Tables 4 and 5 show two examples of applying a relatively simple re-distribution scheme to the mesh points in the global H and Ω domains, respectively. We can see that although the resulting value of $\max_s(N^I_s + N^{O_c}_s)$ is dramatically reduced in comparison with $\max_s N_s$, the load balancing quality can be further improved. Regarding Table 5, the reason for only a slight decrease in WT_{CG} is because the portion of the duplicated computations inside the PDE part is relatively small. Figure 7 depicts the distribution of the values of $N^{O_n}_{\Omega_s}, N^{O_c}_{\Omega_s}, N^I_{\Omega_s}$ after applying the simple re-distribution scheme to the Ω mesh points, where $N_\Omega = 919,851$. We can see that the load balancing situation is considerably improved with respect to $N^{O_c}_{\Omega_s} + N^I_{\Omega_s}$, compared with N_{Ω_s}. However, the disjoint re-distribution can clearly be further improved.

Another possibility of overhead reduction lies in the coarse grid corrections within the additive Schwarz iterations, when the global $\mathcal{G}_{\Omega,0}$ mesh has a relatively large number of points. As mentioned in Section 4.3, parallel coarse grid corrections should in such a case replace serial coarse grid corrections. Table 6 shows the results associated with the parallel PDE solver when $N_H + N_\Omega = 1,552,283$ for the global fine mesh level and $N_{\Omega,0} = 28,283$ for the global coarse mesh level. (We note that coarse grid corrections are not used in the additive Schwarz iterations for treating the $(1,1)$ diagonal block of $\mathcal{D}$.) Both the serial and parallel coarse grid solvers use CG iterations without preconditioning on the level of the global $\mathcal{G}_{\Omega,0}$ mesh. The reason for a deteriorated performance for $P = 2$ in Table 6 is that the overhead in the parallel

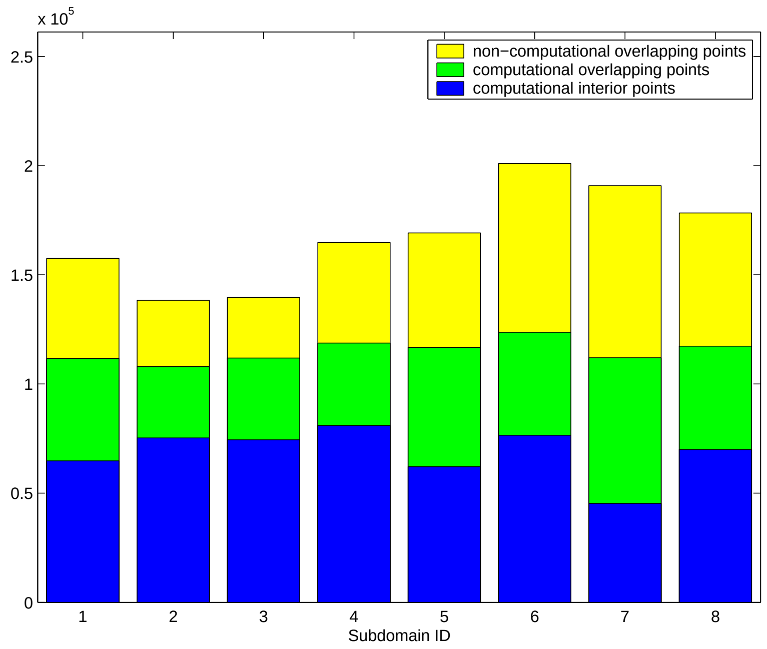

Fig. 7. The effect of applying a disjoint re-distribution to the Ω mesh points, where $N_\Omega = 919,851$ and the number of subdomains is 8.

Table 6. The effect of using parallel coarse grid corrections (CGCs) instead of serial CGCs. The total number of unknowns at the global fine mesh level is $N_H + N_\Omega = 1,552,283$, and WT_{CG} denotes the time consumptions for one time step on a Linux cluster; see Section 6 for more details.

	Serial CGCs	Parallel CGCs
P	WT_{CG}	WT_{CG}
2	42.63	44.82
4	26.42	26.23
8	20.86	19.42
16	17.21	13.46

coarse grid solver exceeds the gain from parallelization. The advantage of a parallel coarse grid solver thus becomes visible when P is large.

6 Numerical Experiments

In this section, we will report some more numerical experiments of the electro-cardiac simulation, where we have used the realistic global domains H and Ω as depicted in Figure 1. The coarsest global $\mathcal{G}_{H,0}$ mesh consists of 56,568 tetrahedral elements and 11,306 mesh points, while the coarsest global $\mathcal{G}_{\Omega,0}$

mesh consists of 162,120 tetrahedral elements and 28,283 mesh points. Since both the global coarsest meshes have quite high resolution, we directly partition the $\mathcal{G}_{\Omega,0}$ mesh (without adaptive refinement) into overlapping subdomain $\mathcal{G}_{\Omega_s,0}$ meshes, $1 \leq s \leq P$. We remark that our mesh partitioning scheme uses the Metis [14] software package to first generate intermediate non-overlapping subdomain meshes, which are then expanded to introduce a certain amount of overlap between neighboring subdomains. Depending on the desired resolution of the finest subdomain meshes, we adaptively refine the $\mathcal{G}_{\Omega_s,0}$ subdomain mesh J_s times on each subdomain s, as described in Section 4.2. We remark that the subdomain mesh hierarchy $\{\mathcal{G}_{H_s,j}\}$ $(0 \leq j \leq J_s)$ is obtained by "cutting out" the portion of the subdomain mesh hierarchy $\{\mathcal{G}_{\Omega_s,j}\}$ that lies inside H.

We have chosen the Winslow cell model (see [36]) for the numerical experiments in this section. Regarding the anisotropic conductivity tensors M_i and M_e, which are defined in (4), the following σ values are used:

$$\sigma_i^l = 3.0, \quad \sigma_i^t = 0.31525, \quad \sigma_i^n = 1.0, \tag{23}$$

$$\sigma_e^l = 2.0, \quad \sigma_e^t = 1.3514, \quad \sigma_e^n = 1.65. \tag{24}$$

Moreover, the vectors $\mathbf{a}_l$ and $\mathbf{a}_n$ are depicted in Figure 2. For the surface-to-volume scaling factor used in (1), we have chosen $\chi = 5.0 \times 10^{-4}$.

When the global 2×2 block system (14) is solved by the preconditioned parallel CG iterations at each time step, we consider that the convergence is reached when the global residual vector is reduced by a factor of 10^4 from its initial value at the beginning of the time step, measured in the L_2-norm. For the multigrid V-cycles that are used as the subdomain solvers in the additive Schwarz iterations, the pre- and post-smoothers are chosen as three SOR iterations for the subdomain mesh levels $1 \leq j \leq J_s$. On the level of the coarsest subdomain mesh $\mathcal{G}_{H_s,0}$ or $\mathcal{G}_{\Omega_s,0}$, 20 SSOR iterations are used. Coarse grid corrections have been used in association with the additive Schwarz iteration handling the second diagonal block in $\mathcal{D}$. The chosen global coarse grid solver on the $\mathcal{G}_{\Omega,0}$ mesh level is also a parallel CG solver, which aims to reduce the associated residual vector by a factor of 10. We remark that for the additive Schwarz iteration handling the first diagonal block in $\mathcal{D}$, experiments show that coarse grid correction is not necessary.

The parallel electro-cardiac simulator is programmed in the scientific computing environment of Diffpack; see [15, 10]. The implementation of the additive-Schwarz block preconditioner uses the object-oriented programming techniques and a generic framework for overlapping domain decomposition algorithms; see e.g. [4, 16].

In Table 7, we list the wall-time measurements of four different tasks within one time step:

1. WT_{ODE} denotes the wall-time consumption of the ODE part.
2. WT_{discr} denotes the wall-time consumption of the finite element discretization procedure for building the 2×2 block system (14). We note that

Table 7. The wall-time measurements (in seconds and obtained on an Itanium-cluster) of different tasks within one time step during parallel full-scale electro-cardiac simulations.

J_s	N_H	N_Ω	P	WT_{ODE}	WT_{discr}	I_{CG}	WT_{CG}	WT_{step}
1	82,768	219,398	1	52.85	41.08	15	28.30	153.19
			2	33.42	24.26	7	13.52	93.44
			4	18.98	13.20	7	8.25	53.64
			8	10.26	7.55	7	4.63	30.55
			16	6.20	4.00	8	3.89	23.14
2	632,432	919,851	1	381.58	246.65	16	134.93	982.11
			2	241.85	153.36	7	44.82	585.88
			4	133.45	84.31	7	26.23	327.59
			8	75.21	49.78	8	19.42	195.79
			16	47.77	26.18	8	13.46	127.79
3	4,942,624	5,762,729	1	N/A	N/A	N/A	N/A	N/A
			2	N/A	N/A	N/A	N/A	N/A
			4	1518.95	676.02	9	196.51	2511.18
			8	942.48	383.61	10	121.36	1534.65
			16	617.41	191.27	10	78.19	952.51

WT_{discr} is noticeably larger for the first time step (which is reported in Tables 7 and 8) than for the subsequent time steps, because both $\mathcal{A}$ and the right-hand side vector in (14) need to be built in the first time step, whereas only the right-hand side vector needs to be updated in the subsequent time steps.

3. WT_{CG} denotes the wall-time consumption of the preconditioned parallel CG iterations for solving (14).
4. WT_{step} denotes the total wall-time consumption of one time step.

The measurements in Table 7 are obtained on a Linux cluster that consists of 1.3 GHz Itanium2 processors, inter-connected through a Gigabit ethernet. We can observe that the number of the parallel CG iterations, which is denoted by I_{CG}, remains independent of both the number of degrees of freedom and the number of subdomains (when $P \geq 2$). In fact, the parallel additive-Schwarz block preconditioner has better convergence properties than the serial multigrid block preconditioner. For $J_s = 3$, the finest meshes contain so many points that the simulator can not be run on fewer than four processors. Therefore, measurements for $P = 1$ and $P = 2$ are absent for $J_s = 3$ in Table 7.

In Table 8, we list the corresponding measurements that are obtained on an SGI Origin 3800 machine. Due to a faster communication network on the SGI machine (and slower processors), the scalability of the measurements should in general be better than that of Table 7. However, due to the extremely heavy work load on the SGI machine, some of the measurements in Table 8 have been “slowed down” quite considerably and are thus not very accurate.

Table 8. The wall-time measurements (in seconds and obtained on an SGI Origin 3800 machine) of different tasks within one time step during parallel full-scale electro-cardiac simulations.

J_s	N_H	N_Ω	P	WT_{ODE}	WT_{discr}	I_{CG}	WT_{CG}	WT_{step}
1	82,768	219,398	8	19.91	20.68	7	9.07	55.30
			16	16.12	10.84	8	4.70	33.25
			32	9.13	5.65	8	3.47	21.70
2	632,432	919,851	8	155.63	169.46	8	27.30	385.08
			16	121.67	83.76	8	16.37	242.93
			32	68.59	41.65	8	15.45	142.51
3	4,942,624	5,762,729	8	1233.72	1481.66	9	386.63	3406.51
			16	949.98	755.67	10	286.89	2206.67
			32	534.43	381.01	10	144.98	1197.09

7 Concluding Remarks

Building a high-performance parallel electro-cardiac simulator relies mostly on an efficient parallel solver for the involved PDEs. Numerically, the convergence speed of the parallel CG iterations needed in the PDE part is ensured by a matching parallel substitute of the serial multigrid block preconditioner $\mathcal{D}$ defined in (16). That is, two separate additive Schwarz iterations act as approximate solvers for the two diagonal blocks in $\mathcal{D}$, while also incorporating coarse grid correction and using multigrid V-cycles as the subdomain solvers. With respect to the parallel implementation, satisfactory performance can be obtained when attention is paid to a good work load balance and the reduction of unnecessary duplicated computations. However, due to the complicated shape of the realistic three-dimensional heart and torso meshes, finding a well balanced set of subdomain mesh hierarchies while restricting the volume of communication remains a challenging task. This issue should be further investigated in the future work.

Acknowledgement

The work presented in this chapter has been partially supported by the Norwegian Research Council (NFR) through *Programme for Supercomputing* in form of a grant of computing time.

References

1. W. L. Briggs, V. E. Henson, and S. F. McCormick. *A Multigrid Tutorial.* SIAM, 2nd edition, 2000.
2. A. M. Bruaset. *A Survey of Preconditioned Iterative Methods.* Pitman Research Notes In Mathematics Series 328. Longman Scientific & Technical, 1995.

3. M. L. Buist and A. J. Pullan. The effect of torso impedance on epicardial and body surface potentials: A modeling study. *IEEE Trans. Biomed. Eng.*, 50(7):816–824, 2003.
4. X. Cai. Domain decomposition in high-level parallelization of PDE codes. In C-H. Lai et al., editor, *Domain Decompostion Methods in Science and Engineering*, pages 382–389. Domain Decomposition Press, 1999.
5. X. Cai. Overlapping domain decomposition methods. In H. P. Langtangen and A. Tveito, editors, *Advanced Topics in Computational Partial Differential Equations – Numerical Methods and Diffpack Programming*, pages 57–95. Springer, 2003.
6. X. Cai and G. T. Lines. Enabling numerical and software technologies for studying the electrical activity in human heart. In J. Fagerholm et al, editor, *Applied Parallel Computing - Advanced Scientific Computing, 6th International Conference, PARA 2002*, number 2367 in Lecture Notes in Computer Science, pages 3–17, Espoo, Finland, 2002. Springer-Verlag.
7. X. Cai, G. T. Lines, and A. Tveito. Parallel solution of the Bidomain equations with high resolutions. In *Proceedings of the Parco03 Conference (to appear)*, Dresden, Germany, 2004. Elsevier Science.
8. X. Cai and K. Samuelsson. Parallel multilevel methods with adaptivity on unstructured grids. *Computing and Visualization in Science*, 3:133–146, 2000.
9. T. F. Chan and T. P. Mathew. Domain decomposition algorithms. In *Acta Numerica 1994*, pages 61–143. Cambridge University Press, 1994.
10. Diffpack Home Page. http://www.diffpack.com.
11. P. C. Franzone and L. F. Pavarino. A parallel solver for reaction-diffusion systems in computational electrocardiology. *Mathematical Models and Methods in Applied Sciences*, 14(6):883–911, 2004.
12. I.J. Le Grice, P.J. Hunter, and B.H. Smaill. Laminar structure of the heart: a mathematical model. *Am. J. Physiol. Heart. Circ. Physiol.*, 272:H2466–H2476, 1997.
13. W. Hackbusch. *Multigrid Methods and Applications.* Springer, Berlin, 1985.
14. G. Karypis and V. Kumar. Metis: Unstructured graph partitioning and sparse matrix ordering system. Technical report, Department of Computer Science, University of Minnesota, Minneapolis/St. Paul, MN, 1995.
15. H. P. Langtangen. *Computational Partial Differential Equations - Numerical Methods and Diffpack Programming.* Texts in Computational Science and Engineering. Springer, 2nd edition, 2003.
16. H. P. Langtangen and X. Cai. A software framework for easy parallelization of PDE solvers. In *Proceedings of the Parallel Computational Fluid Dynamics 2000 Conference*, 2001.
17. G. T. Lines. *Simulating the Electrical Activity in the Heart - A Bidomain Model of the Ventrciles Embedded in a Torso.* PhD thesis, Department of informatics, University of Oslo, 1999.
18. G. T. Lines, M. L. Buist, P. Grøttum, A. J. Pullan, J. Sundnes, and A. Tveito. Mathematical models and numerical methods for the forward problem in cardiac electrophysiology. *Computations and Visualization in Science*, 5:215–239, 2003.
19. G. T. Lines, X. Cai, and A. Tveito. A parallel solution of the bidomain equations modeling the electrical activity of the heart. Technical report, Simula Resserch Laboratory, 2001.
20. C. H. Luo and Y. Rudy. A dynamic model of the cardiac ventricular action potenial. *Circulation Research*, 74:1071–1096, 1994.

21. K.-A. Mardal and A. Tveito. Optimal preconditioners for discrete versions of the bidomain model. *Submitted to: SIAM J. Sci. Comp.*, 2004.
22. M. Murillo and X.-C. Cai. A fully implicit parallel algorithm for simulating the non-linear electrical activity of the heart. *Numerical Linear Algebra with Applications*, 11:261–277, 2004.
23. M. Pennacchio and V. Simoncini. Efficient algebraic solution of reaction–diffusion systems for the cardiac excitation process. *Journal of Computational and Applied Mathematics*, 145:49–70, 2002.
24. J. B. Pormann, C. S. Henriquez, J. A. Board Jr., D. J. Rose, D. M. Harrild, and A. P. Henriquez. Computer simulations of cardiac electrophysiology. In *Proceedings of the 2000 ACM/IEEE conference on Supercomputing*. IEEE Computer Society, 2000.
25. D. Porras, J. M. Rogers, W. M. Smith, and A. E. Pollard. Distributed computing for membrane-based modeling of action potential propagation. *IEEE Trans. Biomed. Eng.*, 47(8):1051–1057, 2000.
26. H. I. Saleheen and K. T. Ng. A new three dimensional finite-differenc bidomain formulation for inhomogeneous anisotropic cardiac tissues. *IEEE Trans. Biomed. Eng.*, 45(1):15–25, 1998.
27. K. Skouibine and W. Krassowska. Increasing the computational efficiency of a bidomain model of defibrillation using a time-dependent activating function. *Ann Biomed Eng.*, 28(7):772–780, 2000.
28. B. F. Smith, P. E. Bjørstad, and W. Gropp. *Domain Decomposition: Parallel Multilevel Methods for Elliptic Partial Differential Equations*. Cambridge University Press, 1996.
29. J. Sundnes, G. T. Lines, X. Cai, B. F. Nielsen, K.-A. Mardal, and A. Tveito. *Computing the Electrical Activity in the Human Heart*. Book accepted for publication by Springer-Verlag, 2004. 278 pages.
30. J. Sundnes, G. T. Lines, P. Grøttum, and A. Tveito. Electrical activity in the human heart. In H. P. Langtangen and A. Tveito, editors, *Advanced Topics in Computational Partial Differential Equations – Numerical Methods and Diffpack Programming*, volume 33 of *Lecture Notes in Computational Science and Engineering*, pages 401–449. Springer-Verlag, 2003.
31. J. Sundnes, G. T. Lines, and A. Tveito. ODE-solvers for a stiff system arising in the modeling of the electrical activity of the heart. *International Journal of Nonlinear Sciences and Numerical Simulation*, 4(1):67–80, 2003.
32. J. Sundnes, G.T. Lines, K.-A. Mardal, and A. Tveito. Multigrid block preconditioning for a coupled system of partial differential equations modeling the electrical activity of the heart. *Computer Methods in Biomechanics and Biomedical Engineering*, 5(6):397–409, 2002.
33. M.-C. Trudel, B. Dubé, M. Potse, R. M. Gulrajani, and L. J. Leon. Simulation of QRST integral maps with a membrane-based computer heart model employing parallel processing. *IEEE Trans. Biomed. Eng.*, 51(8):1319–1329, 2004.
34. L. Tung. *A Bi-domain model for describing ischemic myocardial D-C potentials*. PhD thesis, MIT, Cambridge, MA, 1978.
35. R. Weber dos Santos, G. Plank, S. Bauer, and E. J. Vigmond. Parallel multigrid preconditioner for the cardiac bidomain model. *IEEE Transactions on Biomedical Engineering*, 51(11):1960–1968, 2004.
36. R. L. Winslow, J. Rice, S. Jafri, E. Marban, and B. O'Rourke. Mechanisms of altered excitation-contraction coupling in canine tachycardia-induced heart failure, II, model studies. *Circulation Research*, 84:571–586, 1999.

37. J. Xu. Iterative methods by space decomposition and subspace correction. *SIAM Review*, 34(4):581–613, December 1992.