FAST STRUCTURED AMG PRECONDITIONING FOR THE
BIDOMAIN MODEL IN ELECTROCARDIOLOGY.

MICOL PENNACCHIO* AND VALERIA SIMONCINI†

Abstract. The electrical activity of the heart may be modeled by a non-linear system of partial differential equations known as the bidomain model. Due to the rapid variations in the electrical field, accurate simulations require a fine-scale discretization of the equations and consequently the solution of large severely ill-conditioned linear systems at each time step. Solving these systems is a major bottleneck of the whole simulation. We propose a highly effective preconditioning strategy for a general and popular 3D formulation of the problem. A theoretical analysis of the preconditioned matrix ensuring mesh independence of the spectrum is also described. Numerical comparisons with state of the art approaches confirm the effectiveness of our preconditioning technique. Finally, we show that an equivalent but less exercised formulation provides the best performance, in terms of CPU time.

1. Introduction. Computer simulations of cardiac electrophysiology are a helpful tool in the study of the bioelectric activity of the heart both in normal and pathological conditions. These simulations are based on a system of two PDEs, named the bidomain system, and are coupled to a set of ODEs describing the membrane kinetics of cardiac cells. Indeed the excitation process in the myocardium is a complex phenomenon characterized by rapid ionic fluxes through the cellular membrane separating the intracellular and the interstitial fluid in the myocardium; see [16, 21, 39] for details.

Due to its time and space multiscale nature, the numerical solution of the bidomain system represents a very intensive computational task. More specifically, because of the rapid variations in the electrical field, sufficiently accurate simulations require a fine-scale discretization of the equations. For realistic geometries this leads to a large number of grid points, at least \(O(10^7)\), and consequently large linear systems have to be solved each time step. To limit the computational cost, adaptive techniques and domain decomposition methods have been developed [10, 22, 31, 34, 54, 60]. Adaptivity in space and time may represent a valid solution to reduce the computational cost of the bidomain system, see, e.g., [6], although until now successful investigations have only been reported for moderate size problems. Earlier bidomain investigations based on finite difference discretizations can be found in [5, 40, 45].

Fully implicit methods in time have been considered in few studies (see e.g. [27, 25, 26]) and require the solution of non-linear systems at each time step. However, most numerical studies now employ semi-implicit methods in time: these allow for wider time steps than explicit schemes, at the cost of dealing with a large algebraic linear system at each time step, whose conditioning considerably worsens as the problem dimension increases, resulting in an unacceptable increase in the computational costs of the whole simulation. In this context, preconditioning is therefore mandatory. The design of computationally effective iterative solvers for such linear systems calls for the construction of efficient preconditioners, see [58] for a detailed overview of the methods. Attempts in the literature range from diagonal preconditioners [47], Symmetric Successive Over Relaxation [33], Block Jacobi preconditioners with incomplete LU factorization (ILU) for each block [10, 57], to multigrid [52, 1, 38, 59, 46] and

*Istituto di Matematica Applicata e Tecnologie Informatiche del C.N.R., via Ferrata, 1 - 27100 Pavia, Italy (micol@imati.cnr.it)
†Dipartimento di Matematica, Università di Bologna, Piazza di Porta S. Donato, 5, I-40127 Bologna, Italy; CIRSA, Ravenna and IMATI-CNR, Pavia, Italy (valeria@dm.unibo.it).
symmetric block structured multigrid [24].

In this work we build upon our former experience with structured preconditioning techniques in the 2D setting (cf. [35, 36]), to propose a highly effective preconditioning strategy for a general and popular 3D formulation of the problem. The approach does not require any extra parameter tuning and relies on public-domain, recently developed algebraic multigrid solvers as building block. The resulting preconditioner is nonsymmetric, however the spectral properties are so favourable that a cheap short-term recurrence method such as BICGSTAB can be effectively employed. We describe a novel theoretical analysis of the preconditioned matrix ensuring mesh independence of the spectrum if appropriate multigrid strategies are used. Numerical comparisons with state of the art approaches confirm the effectiveness of our preconditioning strategy. In addition, we show that an equivalent but less exercised formulation provides the best performance, in terms of CPU time, with a saving of up to 30% of the total computational time with respect to the improvement of the more common formulation. These results envision the possibility to simulate the whole excitation process in a close to real time.

General geometric and algebraic multigrid preconditioning have been already applied to the bidomain system in one of the two formulations we shall adopt, and their effectiveness when compared to other classical methods has been reported; see, e.g., [52, 1, 38, 59, 60, 58]. However, many of these studies rely on the application of multigrid methods only on the elliptic equation of the bidomain system related to the extracellular potential. It was recently shown in [49] that, for the same level of accuracy, solving the coupled system of elliptic-parabolic equations is more efficient than dealing with the two equations separately. On the other hand, we refer to [24] for a recent analysis of (symmetric) block structured algebraic preconditioners more closely related to our solution strategy.

The outline of the paper is as follows. In section 2 we introduce the bidomain system and two of its more commonly used formulations. Space and time discretizations are presented in the same section, leading to the description of the algebraic linear systems to be solved at each time step. By using a unified form for the linear system stemming from the discretization of both formulations, in section 3 we introduce the class of structured preconditioners that we wish to analyze. Section 4 is devoted to the convergence analysis of the employed structured preconditioners. To the best of our knowledge, this provides the most realistic spectrum analysis in the literature for this general nonsymmetric preconditioning strategy. Section 5 discusses some implementation issues, and recalls a recently developed inner-outer procedure which will be compared experimentally with our approach. Finally, Section 6 reports on numerical experiments showing the effectiveness of the block structured algebraic multigrid approach for one formulation, and the effectiveness of using the less common formulation.

Throughout the paper we shall often deal with singular matrices. With a little abuse of notation, we shall denote by $A^{-1}$ the pseudoinverse of the symmetric and singular matrix $A$. Nonetheless, the singularity of the matrix is fully taken into account when dealing with $A^{-1}$.

2. The Bidomain Model. The Bidomain model is the most complete model to simulate the bioelectric activity of the heart (see [17, 32, 56]), and it consists of a nonlinear Reaction-Diffusion (R-D) system of equations for the intra- and extracellular potentials $u_{i}$ and $u_{e}$, coupled through the transmembrane potential $v := u_{i} - u_{e}$. In this model the cardiac muscle is viewed as two superimposed anisotropic continuous
media, intra (i) and extracellular (e), occupying the same volume and separated from each other by the cell membrane. The non-linearity arises through the current-voltage relationship across the membrane which is described by a set of nonlinear ODEs (see [23] for more details).

Cardiac tissue model with fiber architecture and anisotropy. The domain \( \Omega \subset \mathbb{R}^3 \), chosen in this work to represent the left ventricle, is modeled by a family of truncated ellipsoids described by the parametric equations:

\[
\begin{align*}
    x &= a(r) \cos \theta \cos \varphi & \theta_1 \leq \theta \leq \theta_2, \\
    y &= a(r) \cos \theta \sin \varphi & 0 \leq \varphi \leq 2\pi, \\
    z &= c(r) \sin \theta & 0 \leq r \leq 1,
\end{align*}
\]

where \( a(r) = a_1 + r (a_2 - a_1) \), \( c(r) = c_1 + r (c_2 - c_1) \) and \( a_i, c_i, i = 1, 2 \) are given coefficients determining the main axes of the ellipsoid, see Fig. 2.1. The surfaces corresponding to \( r = 0 \) and \( r = 1 \) describe the endo- and epicardium respectively.

Anatomic studies showed that the fiber direction rotates counterclockwise from epicardium to endocardium and that fibers are arranged in sheets, running through the myocardial wall [50, 37]. The fiber structure and the anisotropic properties of the cardiac tissue are modeled by the intra- and extracellular conductivity tensors \( M_i = M_i(x) \) and \( M_e = M_e(x) \) defined as:

\[
M_s(x) = \sigma_i^s \mathbf{a}_i(x)\mathbf{a}_i(x)^T + \sigma_n^s \mathbf{a}_n(x)\mathbf{a}_n(x)^T + \sigma_t^s \mathbf{a}_t(x)\mathbf{a}_t(x)^T \quad s = i, e,
\]

with \( \mathbf{a}_i(x) \) unit vector tangent to the cardiac fiber at a point \( x \in \Omega \), \( \mathbf{a}_t \) orthogonal to the fiber direction in the fiber sheet and \( \mathbf{a}_n \) orthogonal to the sheet. The conductivity coefficients measured along the corresponding directions are \( \sigma_i^s, \sigma_n^s, \sigma_t^s \), \( s = i, e \) respectively. If we have axial isotropy, i.e. same conductivity in both tangential and normal directions, the tensors can be written as:

\[
M_s(x) = \sigma_i^s I + (\sigma_i^s - \sigma_t^s) \mathbf{a}_i(x)\mathbf{a}_i(x)^T, \quad s = i, e,
\]

3
where $I$ is the identity matrix and $\sigma^i_s$, $\sigma^e_s$ for $s = i, e$ are the conductivity coefficients along and across fiber, in the (i) and (e) media, assumed constant with $\sigma^i_s > \sigma^e_s > 0$. In both cases it can be easily verified that $M_{i,e}$ satisfy the following uniform ellipticity condition:

$$\exists \lambda^i_m, \lambda^e_M > 0, \quad \lambda^i_m |\xi|^2 \leq \xi^T M_s(x) \xi \leq \lambda^e_M |\xi|^2 \quad \forall \xi \in \mathbb{R}^3, x \in \Omega, \ s = i, e. \quad (2.3)$$

The fibers rotate intramurally linearly with the depth for a total amount of 120° proceeding counterclockwise from epicardium and endocardium. More precisely, in a condition:

$$\sigma \text{ along and across fiber, in the (i) and (e) media, assumed constant with } \sigma^i_s > \sigma^e_s > 0$$

we consider two different formulations, the $(u_i, v, e)$ and the $(u_i, u_e)$ formulations. For both formulations we will deal with a FEM discretization in space and a semi-implicit scheme in time: implicit for the diffusion term and explicit for the reaction term. In spite of the close relationship between the resulting formulations, the associated linear systems provide different challenges, so that the same general preconditioning strategy may lead to surprisingly different performance on the two formulations. We wish to stress that while the $(v, u_e)$ formulation is well accounted for in the current literature, the $(u_i, u_e)$ formulation has been somehow overlooked. This is mostly due to the fact that $v$ is the quantity of chief interest as well as being directly observable from the single cell to the tissue level through optical mapping [13]. Moreover, formulation $(v, u_e)$ has also been preferred for computational reasons: the parabolic and elliptic equations of the formulation can be solved one after the other by means of a block Gauss–Seidel method, see [1, 33, 58]. Nonetheless, we believe the $(u_i, u_e)$ formulation deserves higher consideration due to its very favourable computational performance, also taking into account that $v$ can be readily recovered as $v = u_i - u_e$.

$(u_i, u_e)$ formulation. The evolution of the intra and extracellular potentials $u_i(x, t)$ and $u_e(x, t)$ and transmembrane potential $v(x, t) = u_i(x, t) - u_e(x, t)$ is described by the following reaction–diffusion system, characterized by two nonlinear parabolic equations coupled through the reaction term $I_{ion}$ with a system of ODE’s describing the evolution of the $N_w$ gating and ion concentration variables $w$:

\[\text{given } I_{app} : \Omega \times ]0, T[ \to \mathbb{R}, \ v_0 : \Omega \to \mathbb{R} \ \text{and } w_0 : \Omega \to \mathbb{R}^{N_w}, \ \text{find } u_i, u_e : \Omega \times ]0, T[ \to \mathbb{R}, \ v = u_i - u_e \ \text{and } w : \Omega \times ]0, T[ \to \mathbb{R}^{N_w} \ \text{such that:}\]

\[\begin{align*}
  c_m \partial_t v - \nabla \cdot M_e \nabla u_i + I_{ion}(v, w) & = I_{app} \quad \text{in } \Omega \times ]0, T[ \\
  -c_m \partial_t v - \nabla \cdot M_e \nabla u_e - I_{ion}(v, w) & = -I_{app} \quad \text{in } \Omega \times ]0, T[ \\
  n^T M_{i,e} \nabla u_i & = 0 \quad \text{on } \Gamma \times ]0, T[ \\
  v(\cdot, 0) & = v_0(\cdot), \ w(\cdot, 0) = w_0(\cdot) \quad \text{in } \Omega
\end{align*}\]

(2.4)

where $\Omega \subset \mathbb{R}^3$ models the heart tissue, $\Gamma = \partial \Omega$, $n$ denotes the outward unit normal to the boundary $\Gamma$ and $I_{app}$ is an applied current used to initiate the process. The
function $I_{ion}$ is a nonlinear function of the transmembrane potential $v$ and of suitable ionic variables denoted by $w$ related to the ionic model chosen. There are additional ordinary differential equations governing the evolution of $v$ and $w$, the analysis of which is beyond the scope of this work. We will refer to (2.4) as the $(u_i, u_e)$ formulation of the bidomain model.

Problem (2.4) admits the following variational formulation:

\[ \begin{align*}
\text{given } & \quad v_0 \in L^2(\Omega), \; w_0 \in L^2(\Omega)^N \quad \text{and} \quad I_{app} \in L^2(\Omega \times [0,T]) \; , \; \text{find} \; u_i, u_e \in L^2(0,T;V) \; \text{and} \; w \in L^2(0,T;L^2(\Omega)^N) \; \text{such that} \; \forall t \in [0,T]: \\
& \quad \begin{cases}
\quad c_m \frac{d}{dt} (v, \varphi_i) + a_i (u_i, \varphi_i) + (I_{ion}(v,w), \varphi_i) = (I_{app}, \varphi_i) & \forall \varphi_i \in V \\
\quad -c_m \frac{d}{dt} (v, \varphi_e) + a_e (u_e, \varphi_e) - (I_{ion}(v,w), \varphi_e) = -(I_{app}, \varphi_e) & \forall \varphi_e \in V
\end{cases}
\end{align*} \tag{2.5} \]

with the initial conditions of (2.4), $V = H^1(\Omega)$, $(\psi, \phi) = \int_{\Omega} \psi \phi \; dx \; \forall \psi, \phi \in L^2(\Omega)$ and $a_s(\cdot, \cdot) : V \times V \to \mathbb{R}, s = i, e$, the bilinear symmetric continuous forms defined as

\[ a_s(\psi, \phi) = \int_{\Omega} (\nabla \psi)^T M_s \nabla \phi \; dx, \quad \forall \psi, \phi \in V \quad s = i, e. \tag{2.6} \]

Well posedness analysis of the Bidomain system with FitzHugh-Nagumo ionic model and more general ionic kinetics can be found in [11] and [56] respectively. System (2.4) uniquely determines $v$, while the potentials $u_i$ and $u_e$ are defined only up to the same additive time-dependent constant related to the reference potential, see [11]. To fix such arbitrary constant we impose that $u_e$ has zero average on $\Omega$.

The preconditioners proposed in this paper are mainly related to the structure of the partial differential equations of the bidomain system and are independent of membrane kinetics considered. We do not deal with a specific choice of the ionic model, however, in the numerical results presented in Section 6 we will use the Rogers-McCulloch model, that is:

\[ I_{ion}(v,w) = G v \left( 1 - \frac{v}{v_{th}} \right) \left( 1 - \frac{v}{v_p} \right) + \eta_1 v w, \quad \frac{\partial w}{\partial t} = \eta_2 \left( \frac{v}{v_p} - \eta_3 w \right), \]

where $G, \eta_1, \eta_2, \eta_3$ are positive real coefficients, $v_{th}$ is the threshold potential and $v_p$ the peak potential.

$(v, u_e)$ formulation. The R-D system governing the cardiac electric activity may be written in various forms involving different combinations of the variables $u_e, v; \; v$; see, e.g., [33, 35]. Here we deal with the formulation generally used for the numerical simulations, i.e. with a parabolic equation for the transmembrane potential $v$ coupled with an elliptic equation for the extracellular potential $u_e$:

\[ \begin{align*}
\text{given } & \quad I_{app} : \Omega \times [0,T] \to \mathbb{R}, \; v_0 : \Omega \to \mathbb{R} \text{ and } u_0 : \Omega \to \mathbb{R}^N, \text{ find } v, u_e : \Omega \times [0,T] \to \mathbb{R} \text{ and } w : \Omega \times [0,T] \to \mathbb{R}^N \text{ such that:} \\
& \quad \begin{cases}
\quad c_m \frac{\partial v}{\partial t} - \text{div} \; M_i \nabla v - \text{div} \; M_i \nabla u_e + I_{ion}(v,w) = I_{app} \quad & \text{in } \Omega \times [0,T] \\
\quad -\text{div} \; M_i \nabla v - \text{div} \; (M_i + M_e) \nabla u_e = 0 \quad & \text{in } \Omega \times [0,T] \\
\quad \mathbf{n}^T M_i \nabla v = 0 \quad & \text{on } \Gamma \times [0,T] \\
\quad v(x,0) = v_0(x), \; u_e(x,0) = w_0(x) \quad & \text{in } \Omega.
\end{cases}
\end{align*} \tag{2.7} \]

We will refer to (2.7) as the $(v, u_e)$ formulation of the bidomain model.

The above system differs from problem (2.4) in that the change of variable allows us to replace the second parabolic equation with an elliptic equation, thus loosing the
degenerate temporal structure of system (2.4). This approach was usually preferred because the two equations can be solved one after the other.

The variational formulation of Problem 2.7 reads as follows:

\begin{equation}
\begin{aligned}
given v_0 & \in L^2(\Omega), w_0 \in L^2(\Omega)^N \text{ and } I_{app} \in L^2(\Omega \times [0,T]) &, \text{ find } v, u_e \in L^2(0,T;V) \\
\text{and } w & \in L^2(0,T;L^2(\Omega)^N) \text{ such that } \forall t \in [0,T]: \\
\begin{cases}
c_m \frac{d}{dt}(v,\varphi) + a_i(v,\varphi) + a_i(u_e,\varphi) + (I_{ion}(v,w),\varphi) = (I_{app},\varphi) & \forall \varphi \in V \\
a_i(v,\varphi_c) + a_e(u_e,\varphi_c) + a_i(u_e,\varphi_e) = 0 & \forall \varphi_e \in V
\end{cases}
\end{aligned}
\tag{2.8}
\end{equation}

with the initial conditions of (2.7).

**Space and Time discretization.** Let $T^h$ be a uniform unstructured tetrahedral mesh of $\Omega$ and $V^h$ the associated space of $P^1$ finite elements. We obtain a semi-discrete problem by applying a standard Galerkin procedure. Choosing a finite element basis $\{\varphi_i\}$ for $V^h$, we denote by

\begin{equation}
C = \left\{c_{rk} = \int_{\Omega} \varphi_r \varphi_k \, dx \right\}, \quad A_s = \left\{a_{r,k} = \sum_{K \in T_h} \int_K (A^s \nabla \phi_r)^T M_h \nabla \phi_k \, dx \right\}
\tag{2.9}
\end{equation}

the symmetric mass matrix and stiffness matrices and by $P_{ion}^h$ and $P_{app}^h$ the finite element interpolants of $I_{ion}$ and $I_{app}$, respectively. By using (2.3) and standard finite element arguments, we can assert that $A_s$, $s = i, e$ are symmetric and positive semidefinite with $A_e e = 0$; in particular, $a_{k,k}$ for $k = 1, n$ are positive and $a_{j,k}$ for $j \neq k$.

The time discretization is performed by a semi–implicit scheme (backward Euler method): the diffusion term is treated implicitly while the nonlinear reaction term $I_{ion}$ explicitly. Then, for the $(u_i, u_e)$ formulation (2.4), the following general algebraic system can be obtained:

\begin{equation}
M \xi^{k+1} = b^k \quad \text{with} \quad M = \begin{bmatrix}
C_t + A_i & -C_t \\
-C_t & C_t + A_e
\end{bmatrix},
\tag{2.9}
\end{equation}

with $\xi^{k+1} = [u_i^{k+1}; u_e^{k+1}]$, $v^k = u_i^k - u_e^k$, $b^k = [C_t v^k - I_{ion}^h(v^k, w^{k+1}) + I_{app}^h$, $-C_t v^k + I_{ion}^h(v^k) - I_{app}^h]$, $C_t = \frac{\tau}{\varepsilon} C$ diagonal with positive diagonal entries and $\tau$ the time step. It may be easily verified that $M$ is positive semidefinite (cf., e.g., [33]). Moreover, for $e$ the vector of all ones, $M[e; e] = 0$ and the system is consistent, in that $b$ has zero mean, that is $e^T b = 0$. As in the continuous model, $v^k$ is uniquely determined, while $u_i^k$ and $u_e^k$ are determined only up to the same additive time-dependent constant.

When considering the $(v, u_e)$ formulation (2.7) and using again a finite element discretization in space and a semi–implicit scheme in time, we get:

\begin{equation}
M \xi^{k+1} = b^k \quad \text{with} \quad M = \begin{bmatrix}
C_t + A_i & A_i \\
A_i & A_i + A_e
\end{bmatrix},
\tag{2.10}
\end{equation}

with $\xi^{k+1} = [v^{k+1}; u_e^{k+1}]$, $v^k = u_i^k - u_e^k$ and $b^k = [C_t v^k - I_{ion}^h(v^k) + I_{app}^h; 0]$. It can be easily proven that the symmetric matrix $M$ is positive semidefinite, see [35].

Whichever formulation is chosen for the bidomain system, at each time step a large linear system $M \xi^{k+1} = b^k$ of size $2n \times 2n$ needs to be solved, whose conditioning considerably worsens as the problem dimension increases, resulting in an unacceptable increase in the computational costs of the whole simulation. In this context, preconditioning is therefore mandatory.
Remark 2.1. Classically system (2.10) is solved by a block Gauss–Seidel method involving the two diagonal blocks \([8]\). To allow for accurate solutions the two equations should not be decoupled: an outer iteration has to be employed to reach accurate approximations of the exact solution of the coupled system, see \([8, 33]\). The overall workload is dominated by the solution of the elliptic equation and typically convergence is reached in less than 10 outer iterations. Nonetheless, to increase efficiency, most studies employing this strategy consider the decoupled components independently, and no outer iteration is carried out. On the other hand, it was recently shown experimentally in \([49]\) that the decoupled solution method is less efficient than a coupled scheme for the same level of accuracy. In particular, it was experimentally verified in \([49]\) that the time per iteration is approximately twice as large for the coupled system than it is for the uncoupled one. However, the coupled method is faster overall since it requires fewer iterations per time step to converge with the same accuracy to the solution of the bidomain system.

3. Block Preconditioners. Both formulations are characterized by a \(2 \times 2\) block form, in which all blocks are square and symmetric, while the diagonal blocks are positive (semi)definite. It is therefore natural to derive preconditioners that exploit this structure, as is the case in various multilevel methods (see, e.g., \([48, 53]\)) as well as in saddle point problems \([3]\). On the other hand, our previous numerical experience, see, e.g., \([35, 36]\), showed that the preconditioning strategies for the two considered formulations, \((u_i, u_e)\) and \((v, u_e)\), need to be very different to achieve best performance. In particular, in the \((u_i, u_e)\) formulation best timings were observed when an appropriate preconditioner was used for the whole matrix, that is the block structure was not taken into account. On the contrary, block structured preconditioning was absolutely required for the \((v, u_e)\) formulation. The latter setting is described next in detail.

Symmetric preconditioners. Effective structured symmetric strategies include block diagonal and block factorized preconditioners:

\[
\mathcal{P}_d = \text{blockdiag}(K, D), \quad \mathcal{P}_f = \begin{bmatrix} I & O \\ A_i K^{-1} & I \end{bmatrix} \begin{bmatrix} K & A_i \\ O & D \end{bmatrix},
\]

where \(K\) is a symmetric and positive definite approximation to the (1,1) block \(C_l + A_i\), while \(D\) is a symmetric and positive definite approximation either to the Schur complement or to the (2,2) block \([2]\); we refer to \([35]\) for computational experience with these preconditioners in the 2D setting, and to \([24]\) for further theoretical and numerical discussion in the 3D case. Note that if \(K\) and \(D\) exactly reproduce the (1,1) block and the Schur complement, respectively, then \(\mathcal{P}_{f,\text{ideal}} = \mathcal{M}\). This ideal choice turns out to be computationally unfeasible because too expensive for large problems.

The application of the factorized preconditioner \(\mathcal{P}_f\) corresponds to the solution of two block triangular systems, that is

\[
\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \mathcal{P}_f^{-1} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} K^{-1} & -K^{-1} A_i D^{-1} \\ O & D^{-1} \end{bmatrix} \begin{bmatrix} I & O \\ -A_i K^{-1} & I \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix},
\]

showing that the application of \(\mathcal{P}_f^{-1}\) requires two solves with \(K\) and one solve with \(D\), together with two (cheap) multiplications with the sparse matrix \(A_i\).

In \([35]\) we discussed a-priori convergence results on the convergence of CG when \(\mathcal{P}_d\) or \(\mathcal{P}_f\) are used as preconditioners, for various selections of \(K\) and \(D\). These algebraic results also apply to our 3D setting (cf. section 4).
Non-symmetric preconditioners. The following nonsymmetric “one-sided” version of $P_f$ has been used for nonsymmetric and also for symmetric (indefinite) saddle point problems (see, e.g., [3]):

$$P_M = \begin{bmatrix} K & A_i \\ O & D \end{bmatrix}.$$  

If $K$ and $D$ exactly reproduced the (1,1) block and the Schur complement, respectively, the (nonsymmetric) preconditioned matrix would be the matrix

$$MP_M^{-1} = \begin{bmatrix} I & O \\ A_i(C_t + A_i)^{-1} & I \end{bmatrix},$$

whose spectrum consists of the single unit eigenvalue, so that a minimal residual method such as gmres ([43]) would converge in at most two iterations. In general, the behavior of the approximate versions of $K$ and $D$ is less predictable. The performance of $P_M$ within the indefinite saddle point context highly overcomes its nonsymmetric nature. The situation is considerably different in our context, where the original matrix is positive (semi)definite. Applying $P_M$ destroys symmetry. Remarkably, however, $P_M$ appears to be very competitive with respect to the symmetric choices. First numerical evidence was reported in [35], while a preliminary convergence analysis under somewhat restrictive hypotheses was proposed in [36]. Here we provide an analysis of the spectrum location of the preconditioned problem when $K$ and $D$ are spectrally equivalent approximations to the ideal cases. In particular, if the equivalence scalars (cf. (4.2)-(4.3)) do not depend on the mesh parameters, then we show that the spectral region of $P_M$ is also bounded independently of the mesh parameter.

4. Spectral analysis. In this section we analyze more in detail the spectral properties of the block upper triangular preconditioner described in the previous section. To this end we focus on the $(v, u_e)$ formulation. The analysis of nonsymmetric structured preconditioners with approximation matrices $K$ and $D$ is very hard in general, usually providing only large overestimates of the actual spectrum. Due to the connection between $P_M$ and $P_f$, we are able to more accurately describe the spectral region of the matrix $MP_M^{-1}$.

In the following, the notation $B \leq A$ for two square matrices $B, A$ means that $A - B$ is positive definite; if $A$ and/or $B$ are singular, the notation silently assumes that the relation holds only in the matrix range. Moreover, whenever the inverse of a singular matrix is used, this inverse is meant to be a pseudo-inverse, as all computation is assumed to be performed in the range of the given matrices.

For $K = A_i + C_t$ and $D = A_i + A_e$ (here and in the following the ideal choices), the matrix $MP_M^{-1}$ can be written as:

$$MP_M^{-1} = \begin{bmatrix} I \\ A_i(A_i + C_t)^{-1} \end{bmatrix} \begin{bmatrix} I & O \\ 0 & I \end{bmatrix} \begin{bmatrix} I & O \\ I & S \end{bmatrix}, S = A_i(A_i + C_t)^{-1}A_i(A_i + A_e)^{-1}.$$  

(4.1)

whose real spectrum is bounded independently of the mesh parameter [36].

In the “non-ideal” case, we assume that $K$ and $D$ are spectrally equivalent to the matrices they approximate. More precisely, there exist positive scalars $\alpha_j$ and $\beta_j$, $j = 1, 2$ such that

$$\alpha_1(A_i + C_t) \leq K \leq \alpha_2(A_i + C_t),$$  

(4.2)

$$\beta_1(A_i + A_e) \leq D \leq \beta_2(A_i + A_e).$$  

(4.3)
The bounds above are assumed to hold on the range of the given matrices. Table 4.1 reports rough estimates of these parameters for the example in section 6 for various discretization refinements, when $A_i + A_e$ is approximated by an Algebraic Multigrid method (AMG with the hsl.mpi20 code [4]), while $A_i + C_t$ is approximated by its diagonal. We refer to section 6 for additional information on the actual implementation.

The numbers show a mild dependence on the mesh size in all cases except for $\beta_1$. For the $\alpha_i$ this is expected, since diagonal preconditioning does not fully eliminate the dependence on the mesh. We will see however that such dependence is not harmful. The dependence appearing in $\beta_2$ is slightly more relevant, however this seems to be related to the use of the chosen coarsening strategy (cf. the discussion in section 6).

<table>
<thead>
<tr>
<th>$n$</th>
<th>$\alpha_1$</th>
<th>$\alpha_2$</th>
<th>$\beta_1$</th>
<th>$\beta_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>22,782</td>
<td>0.9668</td>
<td>1.0343</td>
<td>1.0004</td>
<td>20.59</td>
</tr>
<tr>
<td>52,131</td>
<td>0.9468</td>
<td>1.0620</td>
<td>1.0000</td>
<td>15.56</td>
</tr>
<tr>
<td>172,785</td>
<td>0.8981</td>
<td>1.1215</td>
<td>1.0000</td>
<td>22.14</td>
</tr>
<tr>
<td>333,373</td>
<td>0.8526</td>
<td>1.1900</td>
<td>1.0000</td>
<td>21.21</td>
</tr>
<tr>
<td>780,333</td>
<td>0.7865</td>
<td>1.3927</td>
<td>1.0000</td>
<td>30.33</td>
</tr>
</tbody>
</table>

Table 4.1

Values of $\alpha_1, \alpha_2, \beta_1, \beta_2$ for different problem sizes. $D=\text{AMG}(A_i + A_e)$, $K = \text{diag}(A_i + C_t)$.

To proceed with a spectral analysis when $K$ and $D$ are not ideal, we start by summarizing some relevant results from [35] with the current notation. To this end, we recall that the C.B.S. constant $\gamma$ is defined as (see, e.g., [2])

$$
\gamma^2 = \sup_{v \in \mathbb{R}^n \setminus N(A_i + A_e)} \frac{v^T A_i (A_i + C_t)^{-1} A_i v}{v^T A_i v},
$$

and it holds that ([35, Lemma 4.1]) $\gamma^2 \leq (1 + \lambda_{\text{min}}(A_e, A_i))^{-1} < 1$, where $\lambda_{\text{min}}(A_e, A_i)$ denotes the smallest nonzero eigenvalue of $A_e A_i^{-1}$. Due to the properties of the matrices $A_e, A_i$, we obtain

$$
\lambda_{\text{min}}(A_e, A_i) \geq \ell, \quad \ell := \lambda^e_m / \lambda^i_M,
$$

and $\lambda^e_m, \lambda^i_M$ defined in (2.3). If we consider axial isotropy, i.e. $M_{i,e}$ defined in (2.2) and for our choice of data (cf. Table 6.1), we obtain $\ell = (\sigma_i^e - \sigma_i^e)/(\sigma_i^e - \sigma_i^i) = 0.2416$. Therefore, $\gamma^2$ can be easily bounded, independently of the mesh parameter.

Next proposition collects some spectral bounds obtained for $P_{f}^{-1}M$ in [35, Proposition 4.2] (see also [2, Th.9.5]) and [35, Lemma 4.2].

**Proposition 4.1.** Assume that $K$ and $D$ satisfy (4.2) and (4.3), respectively, with $\alpha_2 \geq 1 \geq \alpha_1 > \gamma^2$. Let $\delta_1 = \beta_1$, $\delta_2 = \beta_2 + \alpha_1^{-1}$ and assume that $\delta_2 \geq 1 \geq \delta_1 > \gamma^2$. Moreover, let $\phi(\tau) = \frac{1}{2} (1 + \tau) + \sqrt{\frac{1}{2} (1 - \tau)^2 + 4 \gamma^2}$. Then

$$
\left(1 + \frac{\max\{\alpha_2, \delta_2\} - 1}{1 - \gamma^2} \phi(r_2)\right)^{-1} \leq \lambda(P_f^{-1}M) \leq \left(1 - \frac{1 - \min\{\alpha_1, \delta_1\}}{1 - \gamma^2} \phi(r_1)\right)^{-1},
$$

where $r_2 = \min\{\frac{\alpha_2 - 1}{\delta_2 - 1}, \frac{\delta_2 - 1}{\alpha_2 - 1}\}$ and $\alpha_2 > 1$ and/or $\delta_2 > 1$; and $r_1 = \min\{\frac{1 - \alpha_1}{1 - \alpha_1}, \frac{1 - \delta_1}{1 - \delta_1}\}$ and $\alpha_1 < 1$ and/or $\delta_1 < 1$.

The following result proved in [36] shows that the spectrum of $MP_M^{-1}$ is bounded independently of the mesh parameter for the ideal choice $K = C_t + A_i$ and for judiciously chosen $D$. This result was described in the 2D context and it was a first step.
towards a spectral analysis of the fully approximated preconditioning matrix $P_M$. In fact the result still holds in the three-dimensional case. In this section we complete this analysis for the general case $K \neq C_t + A_i$. To the best of our knowledge this represents the most exhaustive and realistic spectral analysis available in the literature for this general preconditioning strategy with $P_M$.

**Theorem 4.2.** [36, Th. 1] With the previous notation, let $K = C_t + A_i$.

i) If $D = A_i + A_e$, then

$$
\lambda_{\text{min}}(M_P^{-1}_{M,\text{ideal}}) = 1 - \mu, \quad \lambda_{\text{max}}(M_P^{-1}_{M,\text{ideal}}) = 1,
$$

with $\mu \leq (1 + \lambda_{\text{min}}(A_e, A_i))^{-1}$, $\mu$ constant independent of $h$.

ii) If $D$ satisfies (4.3) for some positive $\beta_1, \beta_2$, then either $\lambda(M_P^{-1}_M) = 1$ or $\beta_1(1 - \mu) \leq \lambda(M_P^{-1}_M) \leq \beta_2$, with $\mu$ satisfying the requirement of (i).

We next provide general and insightful bounds for the eigenvalues of the preconditioned matrix resulting from using the preconditioner $P_M$ with spectrally equivalent matrices $K$ and $D$. Our starting point is that the “one-sided” preconditioner can be written as

$$
M_P^{-1} = M_P^{-1} \left[ \begin{array}{cc} I & 0 \\ A_i K^{-1} & I \end{array} \right] .
$$

(4.6)

**Lemma 4.3.** For any symmetric and positive semi-definite $K$ and $D$ we let $P_f = R^T R$, $K = R_K^T R_K$ and $D = R_D^T R_D$ be the Cholesky decompositions of $P_f$, $K$ and $D$, respectively. Setting $H = R^{-T} M R^{-1}$ it holds

$$
R^{-T} M P^{-1} R = H + H \left[ \begin{array}{cc} 0 & 0 \\ R_D^{-T} A_i R_K^{-1} & 0 \end{array} \right] .
$$

(4.7)

*Proof.* The equality follows from multiplying (4.6) by $R^{-T}$ and $R^T$ from the left and right, respectively. \( \square \)

Lemma 4.3 shows a tight connection between the preconditioned matrix $M_P^{-1}_f$ with $P_f$ symmetric and positive semidefinite, and the fully nonsymmetric preconditioned matrix $M_P^{-1}_M$. Keeping in mind that similarity transformations with $R$ do not modify the spectrum, (4.7) reveals that $R^{-T} M P^{-1}_M R$ is a modification of the symmetric matrix $H$, therefore it is possible to bound its eigenvalues by means of the real eigenvalues of $H$ or equivalently, of $M P^{-1}_f$. In the following, $\Im(X) = (X - X^T)/(2i)$ is the skew-Hermitian part of a square matrix $X$.

**Theorem 4.4.** Assume that $K$ and $D$ satisfy (4.2) and (4.3), respectively, for some positive constants $\alpha_j, \beta_j$. Let $\lambda_M$ be an eigenvalue of $M_P^{-1}_M$. Then there exists an eigenvalue $\lambda$ of $M_P^{-1}_f$ such that

$$
|\lambda - \lambda_M| \leq \lambda_{\text{max}}(M_P^{-1}_f) \|R_D^{-T} A_i R_K^{-1}\|, \quad \text{with} \quad \|R_D^{-T} A_i R_K^{-1}\| < \sqrt{\frac{\alpha_1}{\beta_1}},
$$

and

$$
|\Im(\lambda_M)| \leq \|\Im \left( H \left[ \begin{array}{cc} 0 & 0 \\ R_D^{-T} A_i R_K^{-1} & 0 \end{array} \right] \right)\|.
$$
Fig. 4.1. Eigenvalues of $\mathcal{M}_f^{-1}$ (‘o’) and of $\mathcal{M}_M^{-1}$ (‘x’). The imaginary scale is $[-5 \cdot 10^{-3}, 5 \cdot 10^{-3}]$.

Proof. Using Lemma 4.3, the result is an application of Kahan’s theorem [20]; see also [51, sec. IV.5.1]. We are only left to show the bound for $\|R_D^{-T}A_iR_K^{-1}\|$. We have $\|R_D^{-T}A_iR_K^{-1}\|^2 = \lambda_{\text{max}}((R_D^{-T}A_iR_K^{-1})^TR_D^{-T}A_iR_K^{-1})$. Therefore, for any vector $x \neq 0$,

$$
\frac{x^TR_K^{-T}A_iD^{-1}A_iR_K^{-1}x}{x^Tx} = \frac{y^TA_iD^{-1}A_iy}{y^TKy} = \frac{z^TA_i^\frac{1}{2}D^{-1}A_i^\frac{1}{2}z}{z^T A_i^{-\frac{1}{2}} KA_i^{-\frac{1}{2}} z}
$$

$$
= \frac{z^TA_i^\frac{1}{2}D^{-1}A_i^\frac{1}{2}z}{z^Tz} \frac{z^Tz}{z^T A_i^{-\frac{1}{2}} KA_i^{-\frac{1}{2}} z} \leq \frac{1}{\beta_1} \frac{z^TA_i^\frac{1}{2}(A_i+C_l)^{-1}A_i^\frac{1}{2}z}{z^Tz} \frac{z^Tz}{z^T A_i^{-\frac{1}{2}} (A_i+C_l)A_i^{-\frac{1}{2}} z} < \frac{\alpha_1}{\beta_1},
$$

where we used $y = R_K^{-1}x$ and $z = A_i^\frac{1}{2}y$, while the last inequality follows from noticing that $A_i^\frac{1}{2}(A_i+C_l)^{-1}A_i^\frac{1}{2} \leq 1$ and $A_i^{-\frac{1}{2}}(A_i+C_l)A_i^{-\frac{1}{2}} > 1$. $\Box$

<table>
<thead>
<tr>
<th>$n$</th>
<th>22,782</th>
<th>52,131</th>
<th>172,785</th>
<th>333,373</th>
</tr>
</thead>
<tbody>
<tr>
<td>norm</td>
<td>0.2336</td>
<td>0.2512</td>
<td>0.4250</td>
<td>0.4431</td>
</tr>
</tbody>
</table>

Table 4.2 Estimated values of $\|R_D^{-T}A_iR_K^{-1}\|$ for various refinements.

We remark that if the constants $\alpha_1, \beta_1$ are independent of $h$, then $\|R_D^{-T}A_iR_K^{-1}\|$ is bounded independently of $h$. Since all eigenvalues of $\mathcal{M}_f^{-1}$ are bounded independently of $h$, the result of Theorem 4.4 combined with that of Theorem 4.2 implies that the spectrum of $\mathcal{M}_f^{-1}$ is also bounded independently of $h$. The numbers in
Table 4.2 report an estimate of the norm in Theorem 4.4 for various discretizations of the \((v, u)\) formulation. For a much coarser discretization \((n = 1676)\) we were able to explicitly compute the spectra of both \(MP_f^{-1}\) and \(MP_M^{-1}\), and these are reported in Figure 4.1. Notice that the spectrum of \(MP_M^{-1}\) \((\times\) symbol\) remains very close to that of the symmetric counterpart, with an imaginary part that is significantly lower than even \(\lambda_{\text{max}}(MP_f^{-1}) \|R_D^{-T}A_iR_D^{-1}\| \approx 8 \cdot 10^{-2}\).

We conclude this section with an important consideration on the actual expected convergence behavior for a residual minimizing method such as GMRES. It is known that the eigenvalues alone may not give a complete picture of the true behavior of the iterative solver in the nonsymmetric case; information on the associated eigenvectors or invariant subspaces would provide a more reliable picture. In [36] explicit eigenvector bases were given for the ideal and quasi-ideal cases, which could be used for bounding the residual norm of GMRES [43]. In the much more general case analyzed here an explicit expression of the eigenvectors appears cumbersome. A perturbation analysis would allow us to present a more sophisticated though very technical analysis, which is however far beyond the aim of this paper. Nonetheless, we found the eigenvalue distribution to satisfactorily adhere to the actual convergence history of the method, suggesting that the eigenvectors do not play a significant role.

5. Implementation considerations. If the positive definite symmetric preconditioners \(P_d\) or \(P_f\) were employed, the Preconditioned Conjugate Gradient (PCG) algorithm would be the method of choice; this was extensively used, for instance, in the 2D case in [35]. In there, however, we also experimentally observed that \(P_M\) is more effective. This performance is fully confirmed in the 3D case analyzed in this paper (cf. section 6). In this section we describe some important aspects associated with the use of iterative methods and \(P_M\), for various choices of its diagonal blocks.

All block preconditioners mentioned in previous sections require solves with \(K\) and \(D\), once the selection of appropriate matrices or operators is made. In the case of \(y = P_M^{-1}v, y^T = [y_1^T, y_2^T]\) for instance, this entails performing the following two steps:

\[
\begin{align*}
y_2 &= D^{-1}v_2 \\
y_1 &= K^{-1}(v_1 - A_i y_2)
\end{align*}
\]

We recall that \(P_M\) is a one-sided version of \(P_f\), and thus it avoids the extra solve with \(K\) one would have at each application of \(P_f\), lowering the computational cost per iteration. Therefore, given that using \(P_M\) is cheaper than using \(P_f\), the overall effectiveness of the nonsymmetric approach fully relies on the total cost of the employed nonsymmetric solver. Unless the number of iterations is very low, the optimal GMRES method ([44]) may be expensive, with a cost that grows nonlinearly with the number of iterations. On the other hand, due to the very favorable spectral properties of our preconditioned problem (cf. Theorem 4.4), we found the much cheaper BICGSTAB method to perform very satisfactorily [55], with a cost per iteration comparable to that of PCG.

As for the choice of matrices \(K\) and \(D\), we employ an algebraic multigrid preconditioner for \(D\): a preliminary call is made to set up the preconditioner before the actual iterative solution has started; then a call to an AMG routine at each application of the preconditioner is performed. As of \(K\), we found that a simple Jacobi preconditioner, that is \(K = \text{diag}(C_t + A_i)\), was sufficiently effective. We experimentally observed that the constants \(\alpha_i, \beta_i\) in (4.2), (4.3) were only mildly mesh dependent, so that we expect these choices to provide quasi-optimal performance of a method like GMRES. Clearly, the mesh-related optimality of the preconditioned iteration depends on the spectral
equivalence of $D$ and $K$, and not on the choice of the preconditioner structure.

Not unexpectedly, other choices for $K$ and $D$ such as Incomplete Cholesky factorizations did not provide us with mesh independent performance, and in fact their performance significantly degraded with the problem size, therefore these options were soon abandoned.

5.1. Inner-outer procedure. In this section we discuss a different strategy, recently proposed in [14], for approximating the ideal preconditioner $P_{M,\text{ideal}}$. This approach will be compared in section 6 with our strategy to solve for the $(v, u_e)$ formulation. An alternative to Multigrid, Multilevel or Incomplete Decompositions consists of approximating the ideal choices $K = C_t + A_i$ and $D = A_i + A_e$ by means of a preconditioned iterative solver. This approach gives rise to a so called inner-outer procedure: at each iteration of the (outer) solver, two (inner) iterative solves, one with $A_i + A_e$ and one with $C_t + A_i$ are performed. The accuracy at which the inner solvers can be stopped is in general a major concern, requiring the tuning of an extra parameter, the inner stopping tolerance. In [14] the authors state that a very loose stopping tolerance can be used for both inner preconditioned iterations, without significantly influencing the performance of the outer solver. This consideration is particularly crucial for systems with $A_i + A_e$, whose accurate solution would require very many PCG iterations without heavy preconditioning. Instead, solving with $C_t + A_i$ is extremely cheap and not surprisingly the authors report 1-2 inner iterations with it. We also notice that, since the two preconditioning operators change at each iteration, a flexible outer algorithm needs to be employed: The authors of [14] opt for the flexible version of the residual minimizing GMRES method, here and later denoted by FGMRES [42]. In addition to the usual CPU and memory costs of GMRES, the flexible variant requires storing twice as many GMRES long vectors [42].

The use of an inner-outer procedure necessitates of a careful analysis of the final solution accuracy, which is influenced by the interplay between the inner and outer stopping tolerances. We thus proceed with an experimental analysis of the error norm; for the description of the employed data we refer to section 6. As reference solution we used that obtained with the iterative solver by using a very small stopping tolerance (tol=$10^{-12}$); we will denote this solution by $u_{ref}^e$. The solution uniqueness is ensured by the deflation of the null space vector.

The stopping criterion was based on the relative residual norm; experiments with the backward error yielded very similar results. Here and in the sequel, we employ the relative error Euclidean norm, which is commonly restricted to the extracellular potential $u_e$, namely

$$\text{error} = \frac{\|u_{ref}^e - u_e\|}{\|u_{ref}^e\|}, \quad (5.1)$$

(data corresponding to 10 msec after stimulation). The CPU time to reach an equivalent level of accuracy for each method is used as a level of efficiency.

In Table 5.1 we report the error norms for different levels of mesh refinement. We used FGMRES as outer solver, and PCG+ILU$(0)$ as inner solver, as proposed in [14]. We refer to section 6 for additional information on the implementation. The reported digits show that in general the final accuracy depends on the outer tolerance, as expected. However, there are cases where a smaller inner tolerance yields a more accurate outer solution: compare, e.g., the choices tols=$[10^{-5}, 0.12]$ (the numbers denote outer and inner tolerances, respectively) and tols=$[10^{-5}, 10^{-2}]$ for $n = 52 132, 172 785, 333 373$. 13
Table 5.1

<table>
<thead>
<tr>
<th>n</th>
<th>tols = [10^{-5}, 0.12]</th>
<th>tols = [10^{-7}, 0.12]</th>
<th>tols = [10^{-8}, 0.12]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Time</td>
<td>It</td>
<td>error</td>
</tr>
<tr>
<td>22,782</td>
<td>0.09</td>
<td>5</td>
<td>9.00e-3</td>
</tr>
<tr>
<td>52,132</td>
<td>0.19</td>
<td>4</td>
<td>7.35e-2</td>
</tr>
<tr>
<td>172,785</td>
<td>0.87</td>
<td>4</td>
<td>7.71e-2</td>
</tr>
<tr>
<td>333,373</td>
<td>4.52</td>
<td>5</td>
<td>9.53e-3</td>
</tr>
<tr>
<td>780,333</td>
<td>13.22</td>
<td>5</td>
<td>2.21e-2</td>
</tr>
<tr>
<td>1,346,895</td>
<td>26.74</td>
<td>5</td>
<td>1.74e-2</td>
</tr>
<tr>
<td>1,841,622</td>
<td>49.87</td>
<td>6</td>
<td>6.46e-3</td>
</tr>
</tbody>
</table>

Moreover, requiring a more accurate inner solution may allow for a looser outer tolerance, compare, e.g., tols=\[10^{-8}, 0.12\] and tols=\[10^{-7}, 10^{-2}\] with n ≤ 780 333, for comparable timings. Overall, it is clear that some trial-and-error procedure is required whenever an inner-outer method is chosen for this application problem.

6. Numerical results. Computational experiments were carried out to compare the different formulations and solvers introduced in the previous sections. The considered domain Ω is a truncated ellipsoidal region representing the left ventricle, see the left top plot of Figure 6.1. The domain Ω was modeled and discretized using Comsol [12].

The propagation was elicited by applying a current pulse of 1.A/cm³ lasting 1 msec; hence a value of I_{app} = 1 is applied to each grid node of the stimulated region. We considered the same parameter calibration used in [8] (cf. Table 6.1).

Moreover, meshes were built on Ω by using the mesh generator of Comsol that creates unstructured meshes made up of tetrahedral mesh elements. We fixed a starting mesh and built the subsequent meshes by decreasing the mesh diameter. The number of considered mesh nodes is 2n where n ∈ \{22 782, 52 132, 172 785, 333 373, 780 333, 1 346 895, 1 841 622\}, with corresponding mesh size h ∈ \{0.2, 0.15, 0.1, 0.08, 0.06, 0.05, 0.045\} cm; the time step τ was chosen to be equal to 5 · 10^{-2} msec. For these choices of time and space steps it is possible to obtain stable and accurate results as shown by the validation carried out in [8]. Here we kept the time step fixed and equal to the time step generally used for simulations, see e.g. [10]; numerical tests for different values of τ and h can be found, e.g., in [24].

In Figure 6.1 proceeding clockwise from the left top we display the coarsest mesh (to allow for a good visualization of the mesh) and typical extracellular potential maps (on a finer mesh) on the boundary of the domain at 10, 30 and 50 msec after stimulation. To show the propagation through the cardiac wall, in Figure 6.2 we plot
extracellular potential maps on a two dimensional section of the ellipsoid for the same time instants, i.e. at 10, 30 and 50 msec after stimulation.

\[ \chi = 1000 \text{ cm}^{-1} \quad G = 1.5 \times 10^{-3} \text{ } \Omega^{-1} \text{ cm}^{-2} \]

\[ c_m = 1. \mu F \text{ cm}^{-2} \quad i_{app} = 1 \text{ A cm}^{-3} \]

\[ v_p = 100 \text{ mV} \quad v_{th} = 13 \text{ mV} \]

\[ \sigma^e_i = 2 \times 10^{-3} \Omega^{-1} \text{ cm}^{-1} \quad \sigma^e_t = 1.3514 \times 10^{-3} \Omega^{-1} \text{ cm}^{-1} \]

\[ \sigma^i_f = 3 \times 10^{-3} \Omega^{-1} \text{ cm}^{-1} \quad \sigma^i_t = 3.1525 \times 10^{-4} \Omega^{-1} \text{ cm}^{-1} \]

Table 6.1
Parameter calibration.

Most reported experiments below correspond to a typical temporal instant in the time step evolution (10 msec after stimulation), thus the right–hand side includes information generated during the previous time steps.

All computations were performed in FORTRAN90 on a 8GB ram, Intel(R)Core(TM) i7 CPU 920 @ 2.67GHz 64 bit. We used the Intel compiler ifort when employing the HSL library, except for the codes using AGMG, for which we used gfortran.

The stopping tolerance for all methods was chosen a posteriori so as to obtain a comparable error norm of $10^{-5}$, as defined in (5.1). This strategy allowed us to make fair comparisons among methods that employ different stopping criteria. The CPU time to reach the required level of accuracy is used as an efficiency measure.

As building block Algebraic Multigrid (AMG) preconditioners we explored both the classical AMG of [41], in the HSL implementation HSL_MI20 routine [4], and the Aggregation-based Algebraic Multigrid (AGMG) Method of Y. Notay ([28],[29]). Default values of the latter method were used in all instances; we set

\text{control\%one\_pass\_coarsen = .true.}

in HSL_MI20, while leaving all other parameters unchanged.

Formulation $(u_t, u_e)$. The Conjugate Gradient (CG) algorithm was used to solve the large linear system (2.9). The singularity of $M$ is not harmful for CG, as long as the system is consistent. The CG algorithm available within the AGMG code was used. As already mentioned, for this formulation an AGMG preconditioner on the whole matrix, hereafter $P_{\text{whole}}$, appeared to be by far the best performing approach among the choices we have tested. This behavior fully confirms our findings in the 2D case, see, e.g., [35, 36], and thus we omit reporting results with other preconditioners. We emphasize that the good performance of this simple unstructured approach is apparently strongly related to the particular sparsity structure of the blocks of $M$ in this formulation (cf. (2.9)). In particular, the presence of negative off-diagonal elements, due to the matrix $-C_t$, and the larger weight of the diagonal entries, seem to make the coarsening procedure highly effective, compared to the $(v, u_e)$ case, where the nondiagonal block $A_t$ includes positive entries (cf. also Table 6.8). The CPU times, number of iterations and associated error norms are displayed in Table 6.2. We start by emphasizing the very low CPU time associated with the solution of the finest discretization, which involves a matrix $M$ of size $2n = 3 683 244$, envisioning the possibility to simulate the whole excitation process in a very affordable time frame. We also notice that both the number of iterations and the relative error norm decrease.

Footnotes:
1. When using AGMG with the ifort compiler we encountered breakdown at run time, therefore gfortran was used.

2. AGMG default values: nsmooth=2,nstep=-1,nlvcyc=0,resi=0.25,checkdd=5.0,trswc=0.25, trspos=0.48, scalcg=1.
Fig. 6.1. Truncated ellipsoid modeling the left ventricle: starting mesh (top-left) and extracel-
lar potential map at 10, 30 and 50 msec after stimulation.

Fig. 6.2. Extracellular potential map on a vertical section of the ellipsoid at 10, 30 and 50
msec after stimulation.

as mesh refinement takes place, providing a favorable “mesh-dependent” performance.

To increase our understanding of the preconditioner performance, we provide estimates for the associated spectral interval, as the problem size increases. Table 6.3 reports estimates for the parameters $\chi_1, \chi_2$ in the inequalities $\chi_1 \mathcal{M} \leq \mathcal{P}_{\text{whole}} \leq \chi_2 \mathcal{M}$. Note that $\chi_2$ decreases with $h$, which is consistent with the favourable decrease of the number of iterations.

We use the following well known estimate for the error $\mathcal{B}$-norm after $j$ CG iterations

$$\|\xi_j - \xi\|_{\mathcal{B}} \leq \left(\frac{\sqrt{\text{cond}(\mathcal{B})} - 1}{\sqrt{\text{cond}(\mathcal{B})} + 1}\right)^j,$$

(6.1)

where $\text{cond}(\mathcal{B}) = \lambda_{\text{max}}(\mathcal{B})/\lambda_{\text{min}}(\mathcal{B})$, in which $\lambda_{\text{max}}(\mathcal{B}), \lambda_{\text{min}}(\mathcal{B})$ are the largest and smallest nonzero eigenvalues of $\mathcal{B}$. For $\mathcal{B} = \mathcal{P}_{\text{whole}}^{-1} \mathcal{M}$, we can estimate $\text{cond}(\mathcal{B})$ as
tol = $10^{-7}$  

<table>
<thead>
<tr>
<th>$P_{\text{whole}}$</th>
<th>Time</th>
<th>Iter</th>
<th>err</th>
<th>Time</th>
<th>Iter</th>
<th>err</th>
</tr>
</thead>
<tbody>
<tr>
<td>22,782</td>
<td>0.47</td>
<td>34</td>
<td>3.25e-4</td>
<td>0.70</td>
<td>52</td>
<td>3.97e-5</td>
</tr>
<tr>
<td>52,132</td>
<td>0.82</td>
<td>26</td>
<td>1.46e-4</td>
<td>1.30</td>
<td>41</td>
<td>1.93e-5</td>
</tr>
<tr>
<td>172,785</td>
<td>2.80</td>
<td>24</td>
<td>1.46e-4</td>
<td>4.13</td>
<td>35</td>
<td>1.13e-5</td>
</tr>
<tr>
<td>333,373</td>
<td>5.16</td>
<td>22</td>
<td>6.44e-5</td>
<td>7.32</td>
<td>30</td>
<td>9.15e-6</td>
</tr>
<tr>
<td>780,333</td>
<td>12.69</td>
<td>20</td>
<td>3.60e-5</td>
<td>16.97</td>
<td>26</td>
<td>5.01e-6</td>
</tr>
<tr>
<td>1,346,895</td>
<td>18.78</td>
<td>17</td>
<td>1.92e-5</td>
<td>24.89</td>
<td>22</td>
<td>2.77e-6</td>
</tr>
<tr>
<td>1,841,622</td>
<td>25.41</td>
<td>16</td>
<td>1.73e-5</td>
<td>34.24</td>
<td>21</td>
<td>2.27e-6</td>
</tr>
</tbody>
</table>

Table 6.2

$(u_i, u_e)$ formulation (2.4). CPU time and number of CG iterations to obtain a reduction of the residual norm by a factor of $10^{-7}$ and $10^{-8}$ for $P_{\text{whole}}$ with AGMG at 10 msec after stimulation.

<table>
<thead>
<tr>
<th>$n$</th>
<th>$\chi_1$</th>
<th>$\chi_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>22,782</td>
<td>0.9368</td>
<td>169.52</td>
</tr>
<tr>
<td>52,132</td>
<td>0.9377</td>
<td>114.01</td>
</tr>
<tr>
<td>172,785</td>
<td>0.9479</td>
<td>87.18</td>
</tr>
<tr>
<td>333,373</td>
<td>0.9634</td>
<td>85.14</td>
</tr>
</tbody>
</table>

Table 6.3

Estimates for the parameters $\chi_1, \chi_2$ in the inequalities $\chi_1 M \leq P_{\text{whole}} \leq \chi_2 M$.

cond($\mathcal{B}$) $\approx \chi_2/\chi_1$, so that the error in the energy norm must be below $10^{-5}$ after at most $j = 73$ iterations on the coarsest grid. Faster convergence is observed in practice.

<table>
<thead>
<tr>
<th>$n$</th>
<th>$\chi_1$</th>
<th>$\chi_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>22,782</td>
<td>0.25</td>
<td>0.86</td>
</tr>
<tr>
<td>52,132</td>
<td>0.42</td>
<td>1.22</td>
</tr>
<tr>
<td>172,785</td>
<td>0.88</td>
<td>3.63</td>
</tr>
<tr>
<td>333,373</td>
<td>1.76</td>
<td>6.63</td>
</tr>
<tr>
<td>780,333</td>
<td>4.25</td>
<td>16.19</td>
</tr>
<tr>
<td>1,346,895</td>
<td>7.78</td>
<td>27.05</td>
</tr>
<tr>
<td>1,841,622</td>
<td>11.36</td>
<td>37.96</td>
</tr>
</tbody>
</table>

Table 6.4

$(u_i, u_e)$ formulation (2.4). Setup time, CPU time and number of CG iterations to obtain a reduction of the residual norm by a factor of $10^{-7}$ for $P_{\text{whole}}(u_i, u_e)$ at 10 msec after stimulation with HSL_MI20 and AGMG.

In Table 6.4 we report a comparison with the AMG code available in the HSL library, the HSL_MI20 routine [4]. The preconditioner was employed within the HSL_MI21 routine implementing the CG method [19]. The table reports the preconditioner setup time, the total elapsed time for the CG relative residual norm to pass the stopping tolerance, the number of CG iterations and the corresponding relative error norm (cf. (5.1)). Stopping tolerances were tuned so as to obtain similar final error
Table 6.5

<table>
<thead>
<tr>
<th>n</th>
<th>$P_D$ mi20 CG tol=$10^{-7}$</th>
<th>$P_M$ mi20 Bicgstab tol=$10^{-7}$</th>
<th>$P_M$ mi20 Gmres tol=$10^{-7}$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Time</td>
<td>It</td>
<td>error</td>
</tr>
<tr>
<td>22,782</td>
<td>0.22</td>
<td>18</td>
<td>1.97e-5</td>
</tr>
<tr>
<td>52,132</td>
<td>0.62</td>
<td>21</td>
<td>1.76e-5</td>
</tr>
<tr>
<td>172,785</td>
<td>2.76</td>
<td>27</td>
<td>1.35e-5</td>
</tr>
<tr>
<td>333,373</td>
<td>5.67</td>
<td>28</td>
<td>1.30e-5</td>
</tr>
<tr>
<td>780,333</td>
<td>17.51</td>
<td>36</td>
<td>1.21e-5</td>
</tr>
<tr>
<td>1,346,895</td>
<td>28.79</td>
<td>34</td>
<td>1.26e-5</td>
</tr>
<tr>
<td>1,841,622</td>
<td>41.06</td>
<td>35</td>
<td>1.24e-5</td>
</tr>
</tbody>
</table>

* Bicgstab requires two preconditioning and matrix-times-vector calls per iteration.

Formulation $(v, u_e)$. We report on solving system (2.10) with the non-symmetric preconditioner $P_M$. For the sake of completeness we also report numerical experiments with $P_D$, for the same choices of blocks $D$ and $K$, whereas we leave out the more expensive $P_f$ (cf. section 3). As already mentioned we used BICGSTAB as solver (routine HSL_MI26, see [19]), the diagonal of $A_i + C_t$ as matrix $K$, and the multigrid preconditioner HSL_MI20 as $D$. CPU times (in seconds), number of BICGSTAB iterations and relative error norm, to obtain a reduction of the residual norm by a factor of $10^{-7}$ and an accuracy of order $10^{-5}$ are reported in the second set of columns of Table 6.5. The first set of columns shows the performance of $P_D$, confirming the improvement of up to almost 30% of the nonsymmetric preconditioner $P_M$ over using $P_D$ with the same diagonal blocks. For completeness, the last set of columns of Table 6.5 shows the behavior of the optimal GMRES method (routine HSL_MI24). The digits show that BICGSTAB’s performance is close to that of GMRES, as the number of applications of the preconditioner is only slightly larger; timings are also comparable, while memory requirements are clearly in favor of BICGSTAB, so that the latter will be reported in later comparisons. We also observe a mild dependence of the number of iterations on the problem dimension. According to [4, Example 3.1.1], this could be due to the use of the one-pass coarsening in HSL_MI20, which however provides a significantly faster solution than the use of the standard coarsening. As an example, if we were to use standard coarsening the results in the left set of columns of Table 6.6 would be obtained with $P_M$ applied to GMRES (compare with the last set of columns in Table 6.5). Note that the dependence is now much milder, stabilizing towards 13 iterations; on the other hand, timings are doubled. For completeness, in the right set of columns of Table 6.6 we also show the performance of GMRES when AMG is applied to both $K$ and $D$, instead of diagonal preconditioning for $K$. Results are slightly better than those with diagonal $K$, with some small variability possibly related to the unstructured grid; however the additional computational cost is not

---

3 Each BICGSTAB iteration requires two matrix-vector evaluations and two preconditioning calls.
(v, u_e) formulation (2.7). Left set of columns: HSL MI20 with standard coarsening in HSL MI20 for D. Right set of columns: HSL MI20 with standard coarsening for both blocks (instead of $K = \text{diag}(C_t + A_i)$).

worth the effort. These experiments support our choice of a “light” application of HSL MI20 throughout.

In Table 6.7 we summarize the results for the best performing variants of the discussed methods for both formulations, for a comparable error norm. The inner-outer method of section 5.1 was implemented as follows: the outer method is FGMRES (routine HSL MI15), the inner method is CG (routine HSL MI21), and the preconditioner for both inner solvers is ILU(0) (routine HSL MI11), all from the HSL package [19]. We can see that the computational cost for solving with the inner-outer method is not competitive, with a cost growing significantly on the finest grid, while requiring the tuning of some extra parameter. These experiments suggest that the structured Algebraic Multigrid is superior and more reliable than the presented inner-outer procedure, at least for the proposed setting.

<table>
<thead>
<tr>
<th>$P_M$ MI20 GMRES</th>
<th>$P_M$ MI20 GMRES</th>
</tr>
</thead>
<tbody>
<tr>
<td>Std coars’ng in $D$</td>
<td>Std coars’ng in $D$, $K$</td>
</tr>
<tr>
<td>$n$</td>
<td>Time</td>
</tr>
<tr>
<td>22,782</td>
<td>0.31</td>
</tr>
<tr>
<td>52,132</td>
<td>0.87</td>
</tr>
<tr>
<td>172,785</td>
<td>3.99</td>
</tr>
<tr>
<td>333,373</td>
<td>8.31</td>
</tr>
<tr>
<td>780,333</td>
<td>24.52</td>
</tr>
<tr>
<td>1,346,895</td>
<td>43.80</td>
</tr>
<tr>
<td>1,841,622</td>
<td>61.74</td>
</tr>
</tbody>
</table>

Table 6.6

Summary of performance for $P_{\text{whole}}$ for the $(u_i, u_e)$ formulation, $P_M$ Bicgstab and $P_M$ Fgmres for the $(v, u_e)$ formulation. CPU time, number of iterations and final error norms.

<table>
<thead>
<tr>
<th>$P_{\text{whole}}(u_i, u_e)$ AGMG</th>
<th>$P_M$ MI20 Bicgstab</th>
<th>$P_M$ ILU(0) Fgmres</th>
</tr>
</thead>
<tbody>
<tr>
<td>$n$</td>
<td>Time</td>
<td>It</td>
</tr>
<tr>
<td>22,782</td>
<td>0.47</td>
<td>34</td>
</tr>
<tr>
<td>52,132</td>
<td>0.82</td>
<td>26</td>
</tr>
<tr>
<td>172,785</td>
<td>2.80</td>
<td>24</td>
</tr>
<tr>
<td>333,373</td>
<td>5.16</td>
<td>22</td>
</tr>
<tr>
<td>780,333</td>
<td>12.69</td>
<td>20</td>
</tr>
<tr>
<td>1,346,895</td>
<td>18.78</td>
<td>17</td>
</tr>
<tr>
<td>1,841,622</td>
<td>25.41</td>
<td>16</td>
</tr>
</tbody>
</table>

Table 6.7

The results in the same table also show that for a comparable final accuracy, preconditioning with $P_{\text{whole}}$ the $(u_i, u_e)$ formulation provides the fastest solution. Finally, we observe that all approaches are only mildly mesh dependent, as the number of iterations does not appreciably increase with the mesh refinement.
We remark that AMG on the whole matrix (preconditioner $P_{\text{whole}}$) is not competitive on the $(v, u_e)$ formulation: the number of iterations significantly increases with the problem size, leading to excessive computational costs; a typical performance is reported in Table 6.8. This feature can be related to the particular structure of the matrix in the $(v, u_e)$ formulation, characterized by many rows with non-negative off-diagonal elements associated with the $(1,2)$ and $(2,1)$ blocks (cf. (2.10)).

<table>
<thead>
<tr>
<th>$n$</th>
<th>$P_{\text{whole}}$</th>
<th>tol= $10^{-7}$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Time</td>
</tr>
<tr>
<td>22,782</td>
<td></td>
<td>0.13</td>
</tr>
<tr>
<td>52,132</td>
<td></td>
<td>0.60</td>
</tr>
<tr>
<td>172,785</td>
<td></td>
<td>6.55</td>
</tr>
<tr>
<td>333,373</td>
<td></td>
<td>26.17</td>
</tr>
<tr>
<td>780,333</td>
<td></td>
<td>118.14</td>
</tr>
<tr>
<td>1,346,895</td>
<td></td>
<td>249.44</td>
</tr>
<tr>
<td>1,841,622</td>
<td></td>
<td>299.09</td>
</tr>
</tbody>
</table>

Table 6.8

$(v, u_e)$ formulation. CPU time and number of CG iterations to obtain a reduction of the residual norm by a factor of $10^{-7}$ for $P_{\text{whole}}$ at 10 msec after stimulation.

To complete the set of comparison tests, we also analyze the setup costs and memory requirements for the leading approaches. The setup time for generating the preconditioner is often a bottleneck of multilevel methods. Fortunately, these costs remain affordable, as reported in Table 6.9. A closer look at the table shows that the setup time is not a discriminant among the various methods. Memory requirements seem to differ more remarkably. In Table 6.10 we report the major memory requirements of the used preconditioners: for ILU(0) the computation is performed on the basis of the nonzero elements of the involved matrices; for HSL_Mi20 we considered the nonzero elements of the matrices at different levels of coarsening, i.e. the operator complexity $c_A$ multiplied by the nonzero elements of the original matrix, see [4], formula 11]. We also recall that FGMRES also requires a number of long vectors that is at least twice as the number of performed iterations. As expected, the AGMG preconditioner on the whole matrix (in the $(u_i, u_e)$ formulation) requires the largest amount of memory, followed by that of FGMRES, if inner ILU(0) preconditioning is performed on both blocks $K$ and $D$, as suggested in [14]. Table 6.10 shows that AGMG preconditioning requires roughly the same amount of storage as an ILU(0) preconditioner, but with a much higher performance.

6.1. Overall computational costs in typical simulations. To assess the performance of the various approaches in a more realistic setting, we next report on the results of a longer simulation of the myocardium excitation process. In Table 6.11 we display the total execution times for a simulation of 50 msec, i.e. 1000 time steps, with both formulations. In each case, we only show the best performing setting.

The HSL_Mi20 based preconditioner with BICGSTAB is more efficient than $P_M$ FGMRES, for all considered discretizations.

It is interesting that the performance of all methods is somewhat sensible to the time instant, as the system right-hand side changes significantly throughout the simulation. Indeed, the average time is often lower than that obtained for the single
time step chosen in previous experiments.

For $n = 1,841,622$ and the $(v, u_e)$ formulation, we observe a CPU time reduction of 73% of $P_M$ mi20 bicgstab when compared with $P_M$ fgmres. However, the fastest time is obtained for $P_{\text{whole}}$ with formulation $(u_i, u_e)$, which provides a further 30% CPU time reduction. As already verified, the setup phase time is irrelevant for all methods, compared to the overall cost of the computation.

### 6.2. Comparisons with an alternative formulation

An alternative formulation was recently proposed in [14]. If we linearly combine the two equations of (2.4) with coefficients $\frac{\lambda}{1+\lambda}$ and $\frac{1}{1+\lambda}$, $\lambda > 0$, we obtain the following parameter-dependent non-symmetric formulation of the bidomain system:

\[
\begin{aligned}
\text{find } v, u_e : \Omega \times [0, T] &\rightarrow \mathbb{R} \\
\text{and } w : \Omega \times [0, T] &\rightarrow \mathbb{R}^N \\
\text{such that:}
\end{aligned}
\]

\[
\begin{cases}
\frac{\partial v}{\partial t} - \nabla \cdot \left( \frac{\lambda}{1+\lambda} M_i \nabla v - \frac{\lambda M_e - M_i}{1+\lambda} \nabla u_e \right) + I_{\text{ion}}(v, w) = I_{\text{app}} & \text{in } \Omega \times [0, T] \\
-\nabla \cdot (M_i \nabla v) + (M_i + M_e) \nabla u_e = 0 & \text{in } \Omega \times [0, T] \\
\mathbf{n}^T M_i \nabla v = 0, & \mathbf{n}^T (M_i + M_e) \nabla u_e = 0 \\
v(x,0) = v_0(x), & w(x,0) = w_0(x) \\
\end{cases}
\text{on } \Gamma \times [0, T] \text{ in } \Omega.
\]

Discretization leads to a linear system with coefficient matrix

\[
\mathbf{M} = \begin{bmatrix}
C_t + \frac{\lambda}{1+\lambda} A_i & \frac{\lambda M_e - M_i}{1+\lambda} A_i - \frac{1}{1+\lambda} A_e \\
A_i & A_e
\end{bmatrix}.
\]
This formulation was introduced in [14] to build a non-symmetric preconditioner similar to the one studied here and in [36] for the \((v, u_e)\) formulation.

Table 6.12 reports the number of iterations for two sets of tolerances required by the preconditioner \(P_M\) proposed in [14] applied to the \((v, u_e)\) formulation (2.10), and to the \((\lambda v, u_e)\) formulation (6.2), with two different values of \(\lambda\) as suggested in [15]. We considered the same preconditioner proposed in [14, 15], i.e. a lower triangular preconditioner instead of the upper triangular one chosen in this paper. The preconditioner is implemented by means of an inner-outer method for the diagonal blocks, as described in earlier sections. Since the cost per outer iteration is the same for all methods in the table, the number of iterations represents a good measure of the relative performance.

From the reported digits we can see that the number of iterations even slightly increases when the alternative \((\lambda v, u_e)\) formulation in considered, for both values of \(\lambda\). Since the changes are not really in favor of this parameter-dependent formulation, these experiments strongly suggest that the classical \((v, u_e)\) formulation should be preferred, not last the consideration that no extra tuning of parameters is required.

7. Conclusions. We have shown that (nonsymmetric) block triangular preconditioning provides an efficient tool for solving the large symmetric linear system arising at each time-step in the semi-implicit discretization of a popular formulation of the bidomain model. Efficiency is obtained by using recently developed algebraic multi-grid codes, AGMG and HSL_MIL20, which experimentally show very mild dependence on the mesh, in terms of number of iterations. These AMG methods are building blocks within our structured preconditioner; alternative - public domain - AMG strategies may lead to even better timings of our preconditioning technique. We showed that inner-outer methods are not competitive and may suffer from parameter tuning. We theoretically justified the good performance of the nonsymmetric structured preconditioner by deriving spectral bounds for the preconditioned matrix, which ensure that short-term recurrence system solvers can be efficiently applied.

Finally, our best numerical results were obtained with the less popular but equivalent \((u_i, u_e)\) formulation (2.4), for which the AGMG preconditioned conjugate gradient

\begin{table}[h]
\centering
\begin{tabular}{|c|c|c|c|c|c|c|c|}
\hline
\(n\) & Time & Iter & Time & Iter & Time & Iter \\
\hline
22,782 & 0.29 & 20.94 & 0.21 & 8.59 & 0.22 & 7.12 \\
52,132 & 0.84 & 26.36 & 0.57 & 9.62 & 0.80 & 7.35 \\
172,785 & 2.63 & 22.66 & 2.63 & 12.40 & 4.32 & 8.00 \\
333,373 & 4.69 & 19.38 & 5.32 & 12.55 & 10.96 & 8.01 \\
780,333 & 11.28 & 17.66 & 14.29 & 14.32 & 38.91 & 8.57 \\
1,346,895 & 17.64 & 15.47 & 23.96 & 13.76 & 80.14 & 9.02 \\
1,841,622 & 23.59 & 14.68 & 33.52 & 13.95 & 125.70 & 9.91 \\
\hline
\end{tabular}
\caption{Table 6.11.
Average execution time (in sec) per time step and average iteration count per time step for 50 msec (i.e. 1000 time steps) for \(P_{\text{whole}}\) and the \((u_i, u_e)\) formulation and \(P_M\) HSL_MIL20 BICGSTAB and \(P_M\) FGMRES for the \((v, u_e)\) formulation.}
\end{table}

\textsuperscript{4}Note that in this context, the performance of the upper and lower triangular preconditioners is expected to be very similar.
method alone is particularly effective. We thus encourage the use of this formulation, from which the numerical solutions of the more exercised formulation can be easily recovered.

Acknowledgments. We wish to thank Jennifer Scott, Jonathan Hogg and Milan Mihajlovic for their help with the use of several HSL routines. We thank Yvan Notay for sharing with us his computational experience with the matlab and fortran versions of his AGMG software. We also thank the anonymous referees for their insightful comments.

All reported experiments were carried out on the machines of the SINCEM Laboratory at CIRSA in Ravenna. We are indebted with Luca Giacomelli for technical support.

REFERENCES


