SIAM J. Sci. Comput.
Vol. 32, No. 4, pp. 2395–2417

TWO-LEVEL NEWTON AND HYBRID SCHWARZ PRECONDITIONERS FOR FLUID-STRUCTURE INTERACTION[*]

ANDREW T. BARKER[†] AND XIAO-CHUAN CAI[‡]

Abstract. We introduce and study numerically a two-level Schwarz preconditioner for Newton–Krylov methods for fluid-structure interaction, with special consideration of the application area of simulating blood flow. Our approach monolithically couples the fluid to the structure on both fine and coarse grids and in the subdomain solves, insuring that there is multiphysics coupling during all aspects of the algorithm. The fluid-structure system is discretized on unstructured nonnested meshes, with an overlapping additive domain decomposition on both coarse and fine levels and multiplicative Schwarz preconditioning between levels. We investigate the effect of different coarse discretization sizes, solver stopping criteria, and overlap size, and we demonstrate that the method is robust to physical parameters including the structure's Young's modulus and the timestep size. Finally, we show effective preconditioning of the complicated coupled system, with nearly perfect weak scaling to a thousand processors and millions of unknowns.

Key words. fluid-structure interaction, blood flow, multilevel methods, arbitrary Lagrangian–Eulerian, restricted additive Schwarz, domain decomposition, parallel computing, preconditioning

AMS subject classifications. 65M55, 65M60, 65Y05, 76Z05

DOI. 10.1137/090779425

1. Introduction.

Fluid-structure interaction problems are difficult enough that simulations of realistic phenomena often require parallel processing. In this paper we consider scalable and efficient two-level Newton–Krylov–Schwarz algorithms for monolithic coupling in fluid-structure interaction. The target application is blood flow in arteries, which is a computationally difficult and practically important application area [15, 36]. In particular, the similar densities of blood and artery wall make the coupling between fluid and structure strong in both directions, so that partitioned or iterative procedures have difficulties due to the added-mass effect [1, 8]. Instead of a partitioned procedure, we adopt a monolithic computational approach, coupling fluid to structure in one large system that is solved all at once. This tight coupling allows for robustness to parameters and makes our method immune to the added-mass effect. The resulting system is difficult to solve, but we show here that it can be solved efficiently with effective preconditioning strategies. For more on monolithic coupling, see [5, 16, 21]. Though we focus on the blood-flow problem, our algorithm could be used for more general fluid-structure interaction problems, including ones with deformable structures inside the fluid domain [25].

Besides monolithic coupling in all aspects of the numerical algorithm, the other emphasis of our approach is parallel scalability. Fluid-structure interaction problems in general, and the blood-flow problem in particular, are very computationally demanding and require parallel computing in order to achieve useful resolution and accuracy. In order to use large supercomputers for large problems, we need innovative

[*]Received by the editors December 8, 2009; accepted for publication (in revised form) May 18, 2010; published electronically August 5, 2010. This research was supported in part by the DOE under DE-FC-02-06ER25784, and in part by the NSF under grants CCF-0634894, CNS-0722023, and DMS-0913089.
http://www.siam.org/journals/sisc/32-4/77942.html

[†]Department of Mathematics and Center for Computation and Technology, Louisiana State University, Baton Rouge, LA 70803-4918 (andrewb@math.lsu.edu).
[‡]Department of Computer Science, University of Colorado, Boulder, CO 80309-0430 (cai@cs.colorado.edu).

algorithms, especially preconditioners. Schwarz preconditioning for fluid dynamics problems can be found in [17, 18, 22]. Recent detailed scalability studies of multilevel preconditioners can be found in [18, 28], but as far as we know this kind of study has not been done for the combined fluid-structure interaction system.

In [5], we developed a one-level Newton–Krylov–Schwarz method for fluid-structure interaction that featured monolithic coupling between fluid and structure and good parallel performance. However, for the most physically realistic problems and for large processor counts the scalability of the method deteriorated, because the one-level method does not keep the linear iteration count bounded as we scale the problem. We demonstrate here that adding a coarse grid to the Newton solver and to the additive Schwarz preconditioner to make a two-level method can cause a very large decrease in the number of linear iterations and a corresponding improvement in parallel scalability.

Two-level domain decomposition methods have a long and successful history for preconditioning of various problems. We offer here a very brief survey of some recent developments, especially as they relate to the interest of this paper in incompressible fluid and incompressible elasticity problems. Nonoverlapping domain decomposition theory has been developed for incompressible Stokes equations in [26] and [27], which are related to our problem because they involve domain decomposition methods for saddle-point problems, but in these papers there is no multiphysics coupling and the particular domain decomposition methods are quite different. More closely related is the Schwarz theory for saddle-point elasticity problems developed in [24], which are extended to the incompressible case in [10] and to the time-dependent case in [30]. This last reference is closely related to the present work, in that it presents an analysis of a saddle-point velocity-pressure system preconditioned with an indefinite preconditioner of the same form; that is, smaller saddle-point problems are solved on subdomains and on the coarse level. That is also our approach here, although our problem is nonlinear and couples two saddle-point problems together, while [30] only looks at one, which could be interpreted as either an elasticity problem or a Stokes fluid problem. There is also a careful comparison of different choices of boundary conditions for the subdomain problems in [30].

Two-level Schwarz domain decomposition methods are well understood for elliptic problems. However, it is not as clear how the methods work for coupled problems of mixed type like the fluid-structure interaction system. A complicated coupled problem, different from ours in particulars but similar in terms of coupling, is discussed in [31] and [33]. Here a very ill-conditioned multicomponent time-dependent problem is solved with a multilevel additive Schwarz method, with proofs in [31] of scalability and optimality of the method. Though the equations and application in this study are very different from ours, the results show that we can expect very good results from overlapping two-level Schwarz methods even when the underlying problem is quite complicated. Reference [33] also uses a hybrid preconditioner like ours that is multiplicative between levels and additive within levels, with good numerical results. These studies use static, logically structured meshes for which it is straightforward to find a good coarse mesh and interpolation operators—in contrast, our approach uses moving, completely unstructured meshes.

There are other ways other than Schwarz preconditioners to include a coarse space to speed up solving a linear system, including algebraic multigrid methods [35] and agglomeration methods. Some of these methods show promise, but in this paper we confine ourselves to domain decomposition and particularly Schwarz methods for preconditioning.

In this paper, we investigate numerically several important issues related to the Schwarz preconditioner. One difficulty is how to construct a coarse mesh for a problem on a complicated moving domain, and determining how coarse that mesh should be in order to balance computational cost with preconditioning quality. Other questions include how accurately the coarse problem needs to be solved at each step for a given fine-grid tolerance, what boundary conditions to impose on the subdomain problems, how to interpolate between nonnested unstructured meshes on the different levels, and what effect the overlap parameter has on the fine and coarse grids. We approach our subject by first discussing the equations that we solve and their finite element discretization in section 2, then discussing the two-level Newton and Schwarz methods that give us scalability in section 3, and finally presenting numerical results in section 4. We end the paper with some concluding remarks.

2. Mathematical model and discretization.

We solve the fully coupled and nonlinear equations for fluid-structure interaction with monolithic coupling of the three components: the fluid, the elastic wall structure, and the moving mesh. For the artery wall, we use an incompressible visco-elastic model. We account for the moving fluid domain by using a time-dependent computational mesh in the arbitrary Lagrangian–Eulerian (ALE) framework [11, 14], and we restrict our attention to large blood vessels, where the fluid properties are approximately Newtonian [15], and we model the fluid with the incompressible Navier–Stokes equations. In this section, we present the mathematical formulation of this three-field system.

2.1. A fully coupled model.

Our incompressible linear visco-elastic model for the structure is a very simplistic model for the target application of an artery wall—more sophisticated and realistic models are discussed in [20]. However, it is a fairly reasonable approximation in cases where structural stresses and deformations are quite small.

In their strong form, the equations governing linear elasticity are

$$\rho_s \frac{\partial^2}{\partial t^2}\mathbf{x}_s = \nabla \cdot \sigma_s, \tag{2.1}$$

$$\nabla \cdot \mathbf{x}_s = 0. \tag{2.2}$$

The stress tensor σ_s is given by

$$\sigma_s = -p_s I + (2/3)E_s(\nabla \mathbf{x}_s + \nabla \mathbf{x}_s^T), \tag{2.3}$$

where $\mathbf{x}_s$ and p_s are the solution variables for the structure—the displacement and pressure, respectively—and E_s is the Young's modulus of the incompressible solid. Note that (2.1) can also be written as

$$\rho_s \frac{\partial^2}{\partial t^2}\mathbf{x}_s = -\nabla p_s + (1/3)E_s \Delta \mathbf{x}_s. \tag{2.4}$$

In practice, we include terms so that (2.1) becomes

$$\rho_s \frac{\partial^2}{\partial t^2}\mathbf{x}_s = \nabla \cdot \sigma_s + \beta \frac{\partial}{\partial t}(\Delta \mathbf{x}_s) - \gamma \mathbf{x}_s, \tag{2.5}$$

where β is a visco-elastic parameter. The γ term is included so that we can reproduce a standard fluid-structure test problem as in [2, 29], which uses a reduced one-dimensional structural model of the form

$$\rho_s \frac{\partial^2 \eta}{\partial t^2} - a\frac{\partial^2 \eta}{\partial z^2} + b\eta - c\frac{\partial^3 \eta}{\partial z^2 \partial t} = f, \tag{2.6}$$

where η is the radial displacement of the artery wall, z is the axial coordinate, and a, b, c are parameters.

To specify the grid displacements $\mathbf{x}_f$, we simply use the Laplace equation

$$\Delta \mathbf{x}_f = 0 \tag{2.7}$$

on the interior of the domain, following the practice in [38]. In our numerical simulations, this simple mesh movement scheme performs well, allowing fairly large grid deformations without leading to ill-conditioned elements or adding error to the simulations, but of course many more sophisticated mesh movement schemes can be considered (see [19]).

We model the fluid as a viscous incompressible Newtonian fluid, using the Navier–Stokes equations

$$\frac{\partial \mathbf{u}_f}{\partial t} + (\mathbf{u}_f \cdot \nabla)\mathbf{u}_f + \frac{1}{\rho_f}\nabla p_f = \nu_f \Delta \mathbf{u}_f, \tag{2.8}$$

$$\nabla \cdot \mathbf{u}_f = 0. \tag{2.9}$$

Here $\mathbf{u}_f$ is the fluid velocity vector and p_f is the pressure. The given data include the fluid density ρ_f and the kinematic viscosity $\nu_f = \mu_f/\rho_f$. The above fluid equations are in Eulerian coordinates which, since our domain is moving, need to be modified for the ALE frame. We get

$$\left.\frac{\partial \mathbf{u}_f}{\partial t}\right|_Y + [(\mathbf{u}_f - \omega_g) \cdot \nabla]\mathbf{u}_f + \frac{1}{\rho_f}\nabla p_f = \nu_f \Delta \mathbf{u}_f, \tag{2.10}$$

$$\nabla \cdot \mathbf{u}_f = 0, \tag{2.11}$$

where $\omega_g = \partial \mathbf{x}_f/\partial t$ is the velocity of the moving mesh and Y indicates that the time derivative is to be taken in the ALE frame.

Boundary conditions for the fluid equations typically consist of a Dirichlet condition, where $\mathbf{u}_f$ takes a given profile at the inlet Γ_i, and a zero traction condition

$$\sigma_f \cdot \mathbf{n}_f = \mu_f(\nabla \mathbf{u}_f \cdot \mathbf{n}_f) - p_f \mathbf{n}_f = 0 \tag{2.12}$$

on the outlet Γ_o, where $\mathbf{n}_f$ is the unit outward normal, though we will occasionally use traction inflow conditions as well. Here we have used

$$\sigma_f = -p_f I + \mu_f(\nabla \mathbf{u}_f), \tag{2.13}$$

which is the appropriate variational form for a Neumann boundary condition for the form of the Navier–Stokes equation that we use, even though it is not the physical stress.

The physical system, as well as our model and discretization, has strong coupling between the three fields. At the fluid-structure boundary, we require that structure velocity match fluid velocity,

$$\mathbf{u}_f = \frac{\partial \mathbf{x}_s}{\partial t}, \tag{2.14}$$

which is a generalization of a no-slip, no penetration condition in the case of an unmoving interface. We also enforce that the moving mesh must follow the solid displacement, so that the structure can maintain a Lagrangian description. This condition takes the form

$$\mathbf{x}_f = \mathbf{x}_s. \tag{2.15}$$

Again, this reduces to a homogeneous Dirichlet condition in the case of a rigid wall, which in turn would imply by (2.7) that the ALE mesh is in fact an unmoving Eulerian grid.

The most interesting part of the fluid coupling is the continuity of traction forces at the boundary. This can be written as

$$\sigma_s \cdot \mathbf{n}_s = -\sigma_f \cdot \mathbf{n}_f, \tag{2.16}$$

where $\mathbf{n}_s, \mathbf{n}_f$ are the unit outward normal vectors for the solid and fluid domains, respectively, and σ_s and σ_f are the Cauchy stress tensors for the solid and fluid defined in (2.3) and (2.13), respectively. The condition (2.16) can be thought of as a Neumann-type condition on the structure model.

It is important to emphasize that these coupling conditions are enforced implicitly as part of the monolithic system—they are never enforced as boundary conditions with given data from subproblems, as in the iterative coupling approach.

2.2. A fully coupled finite element discretization. We use the finite element method to discretize the coupled fluid-structure problem in space, using mixed Q_2-Q_1 finite elements for both fluid and structure. As the first step to finite element discretization, we present the weak form of (2.5), (2.2), (2.7), (2.10), (2.11) together with the coupling conditions (2.14), (2.15), (2.16). Find $\mathbf{u}_f, p_f, \mathbf{x}_f, \mathbf{x}_s, p_s$ such that

$$\int_{\Omega_f(t)} \left.\frac{\partial \mathbf{u}_f}{\partial t}\right|_Y \cdot \phi_f + \int_{\Omega_f(t)} [(\mathbf{u}_f - \omega_g)\cdot\nabla]\mathbf{u}_f \cdot \phi_f + \nu_f \int_{\Omega_f(t)} \nabla \mathbf{u}_f : \nabla\phi_f - \int_{\Omega_f(t)} p_f(\nabla\cdot\phi_f) = \int_{\Gamma_t} \mathbf{t}_f \cdot \phi_f,$$

$$\int_{\Omega_f(t)} \psi_f(\nabla\cdot\mathbf{u}_f) = 0, \tag{2.17}$$

$$\int_{\Omega_0} \nabla\xi : \nabla\mathbf{x}_f = 0, \tag{2.18}$$

$$\rho_s \frac{\partial^2}{\partial t^2}\int_{\Omega_s} \mathbf{x}_s \cdot \phi_s + \int_{\Omega_s} \nabla\phi_s : \sigma_s = \int_\Gamma \phi_s \cdot (\sigma_f \cdot \mathbf{n}), \tag{2.19}$$

$$\int_{\Omega_s} \psi(\nabla\cdot\mathbf{x}_s) = 0 \tag{2.20}$$

hold for all test functions $\phi_f, \psi_f, \xi, \phi_s, \psi_s$ in suitable spaces. For more details about the derivation of this weak form and the precise definition of the function spaces involved, see [4].

Discretizing the above weak form in space, we arrive at the semidiscrete form

$$M_f \frac{du_f}{dt} + N(u_f)u_f + K_f u_f - Q_f^T p_f = 0, \tag{2.21}$$

$$Q_f u_f = 0, \tag{2.22}$$

$$\frac{dx_s}{dt} = \dot{x}_s, \tag{2.23}$$

$$M_s \frac{d\dot{x}_s}{dt} + \beta K_s \dot{x}_s + K_s x_s + \gamma M_s x_s + Q_s^T p_s = A_u u + A_p p_f, \tag{2.24}$$

$$Q_s x_s = 0, \tag{2.25}$$

$$K_m x_f = 0. \tag{2.26}$$

Here we represent the solution variables as discrete vectors $u_f, p_f, x_f, x_s, \dot{x}_s$, and we have also reduced the structure equations from second order in time to first order by introducing a structure velocity $\dot{x}_s$, in addition to the structure displacement x_s, in preparation for the time discretization. The matrices A_u and A_p are introduced to account for the traction matching in fluid-structure interaction; that is, they represent (2.16) and correspond to the boundary term in (2.19). The other coupling conditions are included in the monolithic system but are not reflected in the system as written above—they simply identify certain solution variables with each other.

We discretize in time with the second order implicit trapezoid rule $y^{n+1} = y^n + (\Delta t/2)(F^{n+1} + F^n)$. At each timestep we solve a nonlinear system of the form

$$(\tilde{M} - (\Delta t/2)\tilde{K})y^{n+1} = (\tilde{M} + (\Delta t/2)\tilde{K})y^n, \tag{2.27}$$

where

$$y^n = \begin{pmatrix} u_f \\ p_f \\ x_s \\ \dot{x}_s \\ p_s \\ x_f \end{pmatrix}^n, \quad \tilde{M} = \begin{pmatrix} M_f & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & I & 0 & 0 & 0 \\ 0 & 0 & 0 & \rho_s M_s & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}, \tag{2.28}$$

and

$$\tilde{K} = \begin{pmatrix} N(u_f) - \nu_f K_f & -Q_f^T & 0 & 0 & 0 & 0 \\ Q_f & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & I & 0 & 0 \\ A_u & A_p & K_s + \gamma M_s & \beta K_s & -Q_s^T & 0 \\ 0 & 0 & Q_s & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & K_m \end{pmatrix}. \tag{2.29}$$

Since most of the operators above are linear, the Jacobian is just a modification of $\tilde{M} - (\Delta t/2)\tilde{K}$. In particular, we can write the Jacobian as

$$J = \tilde{M} - (\Delta t/2)\tilde{K} + \tilde{Z}, \tag{2.30}$$

where

$$\tilde{Z} = \begin{pmatrix} J_f & 0 & 0 & 0 & 0 & Z_m \\ 0 & 0 & 0 & 0 & 0 & Z_c \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}. \tag{2.31}$$

Here J_f is the derivative from the nonlinear convective term in the momentum equation, while the Z terms represent additional nonlinearity coming from the moving mesh. In the ALE framework, the momentum and continuity equations (2.10), (2.11) depend in a nonlinear way on the fluid mesh displacements $\mathbf{x}_f$. The Z terms in (2.31) above represent the derivatives of this nonlinear dependence.

The $\tilde{M}$ matrix in J is well-conditioned, and $\tilde{K}$ is highly ill-conditioned, but this ill-conditioning is somewhat moderated by the timestep Δt [6]. The Z terms are quite small when Δt is small, but for moderate timestep values, including them is important.

This large, monolithically coupled nonlinear system is solved in one piece by a two-level Newton–Krylov–Schwarz algorithm, which is the key part of this work for insuring parallel scalability, and which we now go on to describe.

3. Two-level Newton and Schwarz methods.

At each timestep (indexed by n) we have to solve the nonlinear system (2.27). We can write it more abstractly as

$$G_h^{n+1}(x_h^{n+1}) = 0, \quad n = 0, 1, \dots . \tag{3.1}$$

We solve this nonlinear system with a two-level Newton–Krylov–Schwarz algorithm—it is this algorithm that is the heart of this paper and the key to achieving parallel scalability and performance.

3.1. Two-level Newton method.

The nonlinear solver is two-level Newton, which simply means that we use our coarse grid to get a good initial guess for solving (3.1). Since Newton's method can be quite sensitive to the initial guess, this is an important part of the algorithm, often reducing the number of Newton solves per timestep significantly. Also, and more interesting, it can in some cases significantly reduce the number of linear iterations needed inside the Newton steps—see section 4.3.

To use the coarse grid to get a good initial guess for (3.1), we first construct an analogous system

$$G_H^{n+1}(x_H^{n+1}) = 0 \tag{3.2}$$

on the coarse grid, with the initial guess for this system given by $x_H^{n+1,0} = I_h^H(x_h^n)$, which is a restriction of the fine-grid solution from the previous timestep—the form and construction of I_h^H will be discussed in section 3.4. Since the fine grid is not in general a refinement of the coarse grid, the nonlinear functional G_H is not a simple restriction of G_h. To solve (3.2) we use an inexact Newton method, so that each Newton step takes the form

$$x_H^{k+1} = x_H^k + s_H^k, \quad k = 0, 1, \dots . \tag{3.3}$$

Here we have dropped the timestep superscripts $n+1$ from the solution vectors for clarity, and instead use k superscripts to identify which Newton step the vector corresponds to. The Newton correction s_H is an inexact solution to a linear system satisfying

$$\left\| G_H(x_H^k) + G_H(x_H^k)'(M_H^k)^{-1} M_H^k s_H^k \right\| \le \max\left\{ \eta_H^r \| G_H^k(x_H^k) \|, \eta_H^a \right\}, \tag{3.4}$$

where $G_H(x_H^k)'$ is the Jacobian matrix for the nonlinear system G_H evaluated at x_H^k (the $n+1$ time level superscripts on G_H have been dropped), $(M_H^k)^{-1}$ is a preconditioner for this linear system (which is reconstructed at every Newton step), and η^r

and η^a are relative and absolute tolerances for the linear solver. Occasionally, instead of (3.4) we will use the preconditioned Richardson iteration

$$x_H^{j+1} = x_H^j + (M_H^k)^{-1}(G_H(x_H^k) - G_H(x_H^k)' x_H^j), \tag{3.5}$$

where j indexes the step of the iteration. This iteration is applied a fixed number of times in place of solving the linear system (3.4) to the specified tolerance. The details of the preconditioner $(M_H^k)^{-1}$ here and its analogue in the fine-grid algorithm make up a key part of our numerical algorithm and will be discussed in detail in section 3.3.

We continue the iteration (3.3) until the convergence criterion given by

$$\|G_H(x_H^k)\| \leq \max\left\{\epsilon_H^r \|G_H(x_H^0)\|, \epsilon_H^a\right\} \tag{3.6}$$

is satisfied, where ϵ^r and ϵ^a are relative and absolute solver tolerances for the Newton method.

To solve the fine-grid nonlinear problem (3.2), we use a similar inexact Newton method with an initial guess given by

$$x_h^{n+1,0} = I_H^h(x_H^{n+1}), \quad n = 0, 1, \ldots, \tag{3.7}$$

and then (dropping the $n+1$ time level superscripts again) use the Newton iteration

$$x_h^{k+1} = x_h^k + s_h^k, \quad k = 0, 1, \ldots, \tag{3.8}$$

where s_h^k is the Newton correction for the kth Newton step and is determined by solving the inexact linear system

$$\left\|G_h(x_h^k) + G_h(x_h^k)'(M_h^k)^{-1} M_h^k s_h^k\right\| \leq \max\left\{\eta_h^r \|G_h^k(x_h^k)\|, \eta_h^a\right\}, \tag{3.9}$$

and we continue the iteration (3.8) until

$$\|G_h(x_h^k)\| \leq \max\left\{\epsilon_h^r \|G_h(x_h^0)\|, \epsilon_h^a\right\} \tag{3.10}$$

is satisfied.

In the next section, we describe the preconditioners M_H^{-1} and M_h^{-1} in turn. M_H^{-1} will be a one-level restricted additive Schwarz preconditioner, while M_h^{-1} will be a hybrid two-level preconditioner that uses something very much like M_H^{-1} as one of its components.

3.2. One-level restricted additive Schwarz. To define the additive Schwarz method, we first partition the finite element mesh on Ω into several meshes on subdomains Ω_ℓ, each one corresponding to a processor in the parallel machine. Then we extend each subdomain Ω_ℓ to overlap its neighbors by a user-specified amount δ, and we denote the overlapping domain by Ω_ℓ'. All of the partitioning, and the overlap extension, are done with respect to elements—since each element contains the same number of unknowns, this procedure balances the load fairly well for the parallel solve.

On each subdomain Ω_ℓ' we construct a subdomain operator $\tilde{B}_\ell$, which is a restriction of the matrix G_H' in (3.4), that is, it contains entries from G_H' corresponding to degrees of freedom contained in the corresponding subdomain Ω_ℓ'. The main work of the preconditioner is doing local solves of the form $\tilde{B}_\ell z_\ell = g_\ell$, where z_ℓ and g_ℓ are restrictions of the solution x and the right-hand side $G(x)$ to a subdomain. The restricted additive Schwarz preconditioner can then be written as

$$M_H^{-1} = (\tilde{R}_1^0)^T \tilde{B}_1^{-1} \tilde{R}_1^\delta + \cdots + (\tilde{R}_N^0)^T \tilde{B}_N^{-1} \tilde{R}_N^\delta. \tag{3.11}$$

The $\tilde{R}_\ell$ operators map a vector defined on the whole space to a vector defined on the subdomain Ω_ℓ or Ω'_ℓ by injection without weighting. Here $\tilde{R}^0_\ell$ is a restriction operator that does not include overlap, while $\tilde{R}^\delta_\ell$ includes the overlap, so in the formulation above the subdomain solve uses information from the overlap region but then does not contribute the computed solution in the overlap region to the global solve [7, 12]. The choice of subdomain solves and restriction and interpolation operators leads to different kinds of Schwarz preconditioners with different properties [37]. In our algorithm the $\tilde{B}_\ell^{-1}$ solves are done with LU factorization, which is expensive, but since the subdomain solve is local to a single processor, the preconditioner is scalable. We follow [30] in imposing Dirichlet boundary conditions to the pressure as well as the velocity in the subdomain solves—this approach shows good scalability and effectiveness and is easy to implement.

3.3. Two-level hybrid preconditioning. Once the coarse nonlinear problem (3.2) has been solved to our satisfaction, we turn to the solution of the fine nonlinear problem (3.1), which is what we need a solution for in order to advance to the next time level. Again, we use a Newton method to solve this, so again we have to solve a linear Jacobian system at each timestep. This linear system, given by (3.9), is solved with flexible GMRES, preconditioned with a two-level hybrid Schwarz preconditioner. Flexible GMRES is a standard iterative method (see, for example, [32]), and is chosen here to give us more flexibility with the various solves inside the preconditioner. In particular, we do not solve the coarse problem exactly, so the effective preconditioner is different at each linear iteration, which leads us to use fGMRES rather than GMRES.

Using two-level Schwarz methods is a well-established domain decomposition preconditioning technique [34, 37]. The basic idea is to facilitate exchange of information between different subdomains, or, equivalently, to efficiently account for smooth components of the error. This is done by including an additional component in the preconditioner, a coarse grid, where computations are cheap but where the smooth global structure of the solution (or of the error) can be represented well. In a one-level approach, if you have N subdomains between subdomain i and subdomain j, then it takes at least N iterations for information to get from i to j. Including a coarse space in the preconditioner can in principle allow the transfer of this information in a single iteration. For some results on multilevel Schwarz preconditioners being applied to fluid dynamics problems, see [18, 22], and for recent work in semiconductor modeling, see [28].

The domain decomposition portion, or fine-grid portion, of the two-level preconditioner is implemented as in [5], and similarly to the coarse solver described in section 3.2. Again, we partition the mesh into mesh subdomains, extend the subdomains so they overlap, and do coarse solves on the mesh subdomains. In analogy to (3.11), this part of the preconditioner can be written

$$M_1^{-1} = \sum_{j=1}^{N} (R_j^0)^T B_j^{-1} R_j^\delta, \tag{3.12}$$

where again the B_j^{-1} are subdomain solves and the R_j are restriction and interpolation operators for the subdomains. The subdomains here are geometrically very similar to the subdomains described in section 3.2, but they contain many more degrees of freedom because here we are partitioning the fine grid—see Figures 3.2 and 3.3.

To include a coarse level in an additive two-level preconditioning approach, we write

$$M_{ad}^{-1} = I_H^h B_0^{-1} I_h^H + \sum_{j=1}^{N} (R_j^0)^T B_j^{-1} R_j^\delta, \tag{3.13}$$

where I_h^H is a restriction from the fine grid to the coarse grid, $I_H^h = (I_h^H)^T$ is the corresponding interpolation operator from coarse grid to fine grid, and B_0^{-1} is a solve on the coarse grid. Different Schwarz algorithms can be devised by different approximate coarse solves, which implies different forms of B_0^{-1}. We will discuss these choices, and also the choices for restriction and interpolation, in sections 3.4 and 3.5.

In our implementation, we combine the coarse-level and fine-level preconditioners multiplicatively, while continuing to use additive Schwarz within the fine level. We can write the application of this hybrid preconditioner M_h^{-1} to a vector x in two steps:

$$z = I_H^h B_0^{-1} I_h^H x, \tag{3.14}$$

$$M_h^{-1} x = z + M_1^{-1}(x - G_h' z) = z + \sum_{j=1}^{N} (R_j^0)^T B_j^{-1} R_j^\delta (x - G_h' z). \tag{3.15}$$

In this hybrid preconditioner, the additive one-level component (3.15) means we can do the local subdomain solves in parallel, while we do the coarse and fine levels sequentially.

The other main considerations in designing a two-level algorithm are formulating the interpolation between the fine and coarse grids, and solving the coarse problem. We now discuss each of these issues in turn.

3.4. Nonnested interpolation and restriction. For interpolation and restriction we construct an interpolation matrix I_H^h which has dimension $n_f \times n_c$, where n_f and n_c are the number of degrees of freedom in the fine and coarse meshes, respectively. We use finite element interpolation—that is,

$$(I_H^h)_{ij} = \phi_j(x_i), \tag{3.16}$$

where ϕ_j is the finite element basis function associated with the jth degree of freedom on the coarse grid and $x_i = (x_i^1, x_i^2)$ is the location of the ith degree of freedom on the fine grid. In practice the value $\phi_j(x_i)$ is also interpolated, since we do not have explicit access to the basis function ϕ_j on an unstructured grid. The matrix I_H^h is considered constant for all three fields, even though the fluid mesh points move—it is expected that the coarse grid moves roughly in the same way as the fine grid, so the same interpolation is still appropriate.

With an unstructured grid, it may occasionally happen that a fine grid point x_f is not contained within any coarse element; see Figure 3.1. In this situation, we do the interpolation for the coarse element that is closest to x_f. This amounts to doing extrapolation and is not very accurate if x_f is far outside the coarse element. However, this occurs rarely (for less than 5 percent of the points in a mesh), and even then x_f is very close to the coarse element. In any case this extrapolation does not degrade the performance of the overall algorithm [9].

3.5. Parallel solution of the coarse linear problem. Both (3.4) and (3.14) are linear problems on the coarse grid, employing the same grid, with the same number of variables, but with different solution operators in general. We use the following method to solve both coarse linear problems.

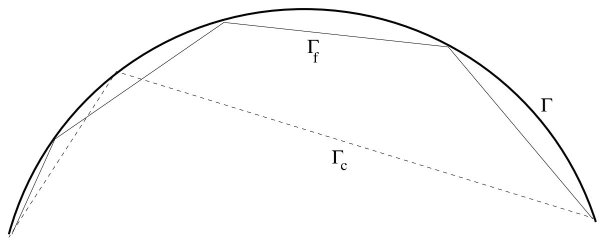

FIG. 3.1. *When meshing a curved physical boundary Γ, it may happen that a point on the fine mesh boundary (Γ_f) is outside all the elements of the coarse mesh whose boundary is given by Γ_c. Our finite elements are quadratic, while the ones pictured here are linear, but the same principle applies.*

What we mean by B_0^{-1} in (3.14) is normally parallel restarted GMRES, preconditioned with a one-level additive Schwarz method, using the same number of subdomains (and therefore processors) as the fine grid. We solve both (3.14) and (3.4) using the one-level algorithm described in section 3.2. However, in order for the coarse grid correction to be helpful we can solve the coarse problem with a much larger error tolerance than the fine problem, saving computational cost while still being an effective preconditioner for (3.14).

Using the same basic algorithm for the one-level method on the coarse grid as on the fine grid has two advantages. First, it is simpler to implement and allows us to reuse some data structures. And second, since we are using the full parallel collective to solve the coarse problem, it allows us to apply the preconditioner multiplicatively, since the coarse solve is done before the fine solve needs any data from it and vice versa. One potential disadvantage is the large number of subdomains on the coarse space, which could lead to the same ill-conditioning problem that drove us to use a two-level method in the first place. In practice, the coarse problem is easy enough to solve and the overlap (which is less costly to increase on the coarse grid) can be made sufficiently large to overcome this difficulty, though for very large simulations we may want to consider additional levels.

The fine and coarse grids in our implementation do not have any necessary connection to each other—they can be generated completely independently by mesh-generating software, and the interpolation and restriction between them is calculated when the simulation runs. In particular, the fine grid is not a refinement of the coarse grid. The fine grid is partitioned for the domain decomposition and parallel processing by Parmetis [23], and the coarse grid inherits that partition—the elements of the coarse grid are assigned to processors that contain nearby fine-grid elements. See Figures 3.2 and 3.3.

The Jacobian matrix $G'_H \approx B_0$ used for linear solves on the coarse grid is not in general a submatrix of the fine-grid Jacobian G'_h, but is instead generated independently on the coarse mesh in precisely the same way as the Jacobian is generated on the fine mesh, and the coarse Jacobian G'_H is assembled for the coarse-grid Newton solve.

4. Numerical results.

In this section we explore the implications of using a two-level Newton–Krylov–Schwarz method and the interplay of various parameters in that method, comparing to the one-level implementation as we go. We do simulations on a straight tube model, where we can verify results found in the literature and more carefully control the mesh size and number of unknowns, and also consider a

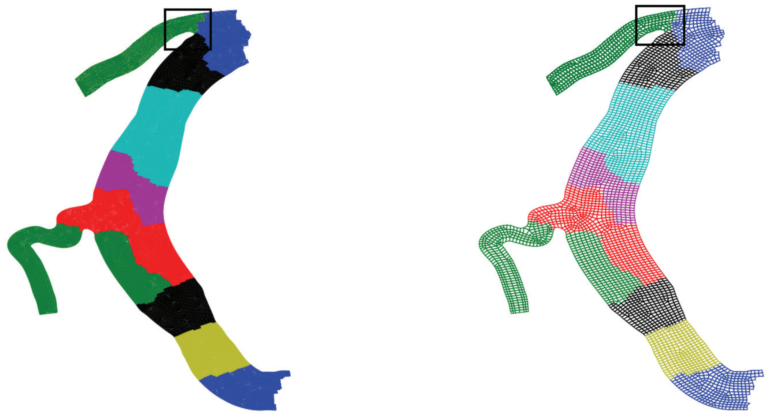

Fig. 3.2. *A portion of the branched artery geometry, with the different subdomains indicated by different shades. A partition of the fine grid is on the left, and a corresponding partition of the coarse grid is on the right. Note that the two partitions are nearly the same. The portions marked with a black box are shown in greater detail in Figure* 3.3.

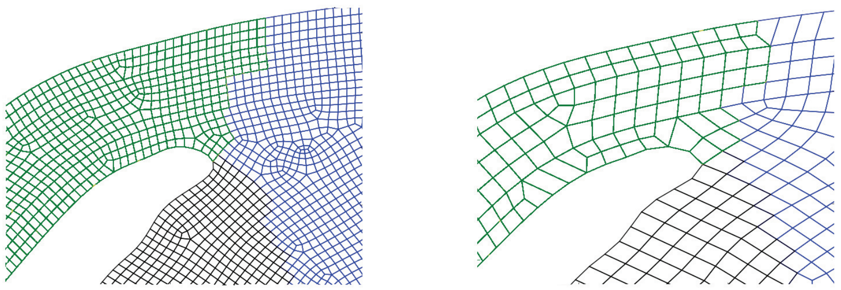

Fig. 3.3. *A portion of the unstructured mesh, where we can see the relative size of the fine (left) and coarse (right) meshes. Note that the fine mesh is not a refinement of the coarse mesh—in our method we can choose a coarse mesh of any size. Note also that the partitions into subdomains are similar for both meshes. A larger portion of this mesh is visible in Figure* 3.2.

more realistic branching artery model derived from clinical data. See [5] for a detailed verification of the same method with a less efficient preconditioner against published data. The solver is implemented using the PETSc libraries [3] and all tests were done on an IBM BlueGene/L supercomputer at the National Center for Atmospheric Research.

In the numerical results in this section, unless otherwise specified, we use an incompressible structure, the fluid density is 1000. kg/m^3, the damping parameter is $\beta = 0.01$, and the kinematic viscosity of the fluid is $\nu_f = 0.0035$ kg/m s.

For the solver parameters, we consider the Newton solver on the fine level to have converged if the (absolute) residual ϵ_h^a is less than 10^{-6}, while for the coarse Newton solver we use a relative tolerance $\epsilon_H^r = 10^{-5}$. For fGMRES on the fine level, we have a (mostly never reached) absolute tolerance of $\eta_h^a = 10^{-14}$ and a relative tolerance η_h^r set by the Eisenstat–Walker method that changes at each iteration, based on the current residual of the nonlinear solve [13]. We restart flexible GMRES every 100 iterations. For the coarse linear solve, we use (nonflexible) GMRES with an absolute tolerance of $\eta_H^a = 10^{-10}$ and a relative tolerance of $\eta_H^r = 10^{-3}$. The coarse linear solver typically converges to its relative tolerance in a few iterations and does not restart.

4.1. Straight tube. The setup of the straight tube problem is taken from [2]. We have a two-dimensional tube 6 cm by 1 cm, with flexible walls at top and bottom of thickness 0.1 cm. A traction condition is applied at the left boundary to induce a pressure pulse, which then travels to the right, deforming the structure as it goes. In this example the Young's modulus $E_s = 7.5 \cdot 10^4$ kg/m s^2, the structure is incompressible and has a density of 1100 kg/m^3, and the inlet pressure pulse takes the form

$$\sigma_f \cdot \mathbf{n}_f = \frac{-P_0}{2}\left[1 - \cos\left(\frac{\pi t}{0.0025 \text{ s}}\right)\right], \tag{4.1}$$

where $P_0 = 2.0 \cdot 10^5$ kg/m s^2. The timestep size is $\Delta t = 0.0001$ s.

The primary motivation for the two-level preconditioner is to improve scalability for the most physically realistic cases, and we demonstrate that scalability in Figures 4.1 and 4.2, which show weak scaling for the straight tube example in the two-grid case, and where the scalability looks very good out to 1024 processors. The linear iterations are kept very nearly constant for the two-level case in sharp contrast to the one-level preconditioner, and the total solution time behaves similarly. The amount of time spent per unknown may seem high, but it is worth emphasizing that this is a computationally expensive problem—each timestep requires several Newton steps, each Newton step contains several GMRES iterations on fine and coarse levels, and the ALE framework requires us to reassemble our matrices at each timestep. Nevertheless, the time spent is reasonable, and in any case the key feature is the parallel scalability, which is very good.

One difficulty in our current approach is that the coarse mesh is solved on the same parallel collective, and by the same method, as the fine mesh. This means that in principle the coarse solver could be subject to the same problem of ill-conditioning as the number of subdomains increases. In most of our simulations, though, the coarse problem is just easy enough that this is not a great consideration. But as we consider scaling our method to larger problems and larger machines, this may be problematic. See Figure 4.3, which shows the increase in coarse solver time for increasing subdomains, which is a result of increasing linear iterations on the coarse level, where we are in effect using a one-level Schwarz method. One possible fix is to simply reduce the number of processors used to solve the coarse problem and let the rest of the processors sit idle during the coarse solve, but this is not the most efficient use of the parallel machine.

A more sophisticated solution would be to add a third or more levels to the preconditioner. This solution would make our approach, which already has much in common with geometric multigrid preconditioners, more like the multigrid method. It might also be possible to use algebraic multigrid to solve the Jacobian systems, but these are complicated coupled systems of mixed type, and standard algebraic multigrid techniques do not work. We note that the kind of weak scaling and performance we

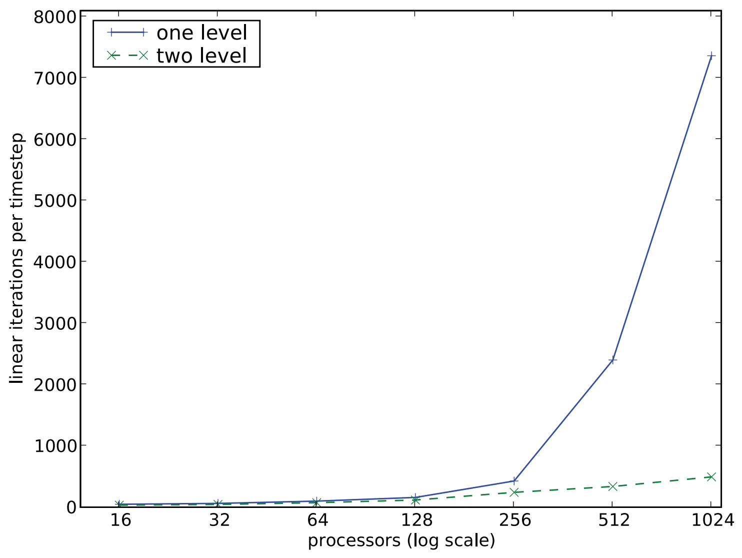

FIG. 4.1. *Weak scaling of the linear solver for a straight tube test problem. The vertical axis shows average number of linear iterations per timestep as we increase the number of subdomains. The addition of a coarse level improves scalability. The number of unknowns increases with the number of processors*—1024 *processors is* $7.1 \cdot 10^6$ *unknowns,* 256 *is* $1.8 \cdot 10^6$*, and so on.*

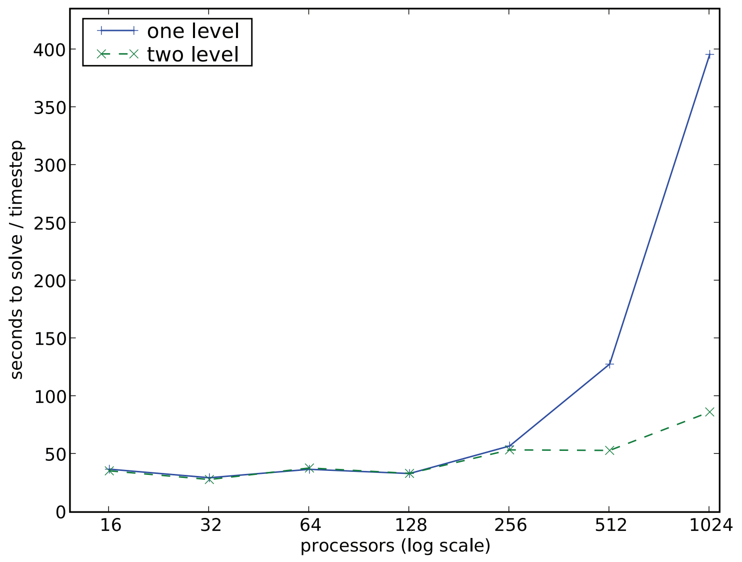

FIG. 4.2. *Weak scaling of total time for a straight tube test problem. The vertical axis shows average walltime in seconds per timestep of the simulation. The addition of a coarse level improves scalability. Again, the number of unknowns increases with the number of processors*—1024 *processors is* $7.1 \cdot 10^6$ *unknowns,* 256 *is* $1.8 \cdot 10^6$*, and so on.*

get is very similar to multilevel methods recently used for the different application of semiconductor modeling [28].

Perhaps the most important implementation detail to consider in designing a two-level method is to choose the size of the coarse grid in order to balance the

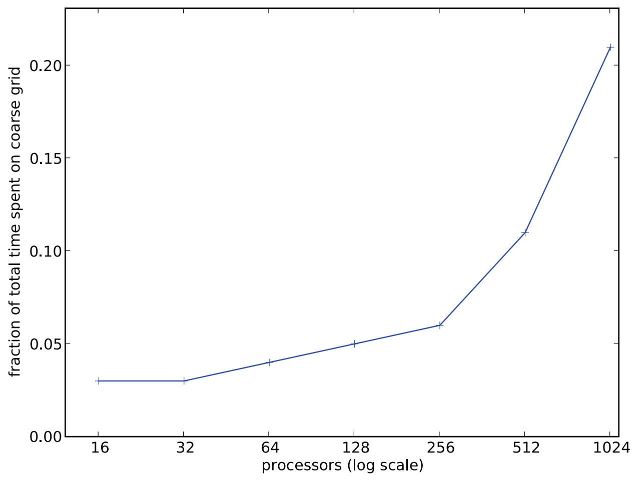

FIG. 4.3. *Weak scaling of the coarse-level solver for the grids and processor counts as in Figures* 4.1 *and* 4.2*. This plot shows the fraction of total compute time spent on the coarse grid—though it is less than a quarter of the time even in the extreme case, it is growing rapidly as the number of subdomains increases.*

improvement in conditioning that comes from using a relatively fine coarse grid with the cost of solving the problem on the coarse grid. In Table 4.1, we present some comparisons of performance and linear iterations for different coarse-grid sizes, while in Table 4.2 we consider the effect of some solver parameters on the computational cost of the coarse solve. In Figure 4.4, the deformation of the straight tube and the corresponding mesh is visible. The deformation, though not very large, is significant, and the good performance of our simple mesh movement scheme is visible.

4.2. Realistic geometry. In this subsection we use a pulmonary artery model taken from clinical data. The geometry for this branching model is visible in Figure

TABLE 4.1
Effect of the coarse grid size on the solver behavior for the straight tube case. The heading "coarse size" reports the number of unknowns on the coarse grid as a fraction of the number on the fine grid, and "coarse time" is the fraction of total compute time spent on the coarse grid.

Unknowns	np	Coarse size	Levels	fGMRES	Coarse time	Walltime
$4.51 \cdot 10^5$	64	0.0	one	74.6	0.00	46.21
$4.51 \cdot 10^5$	64	0.03	two	53.1	0.04	46.33
$4.51 \cdot 10^5$	64	0.12	two	43.0	0.13	46.84
$7.97 \cdot 10^5$	128	0.0	one	123.2	0.00	41.08
$7.97 \cdot 10^5$	128	0.02	two	86.9	0.06	42.87
$7.97 \cdot 10^5$	128	0.07	two	68.7	0.11	43.49
$1.78 \cdot 10^6$	256	0.0	one	313.0	0.00	66.07
$1.78 \cdot 10^6$	256	0.01	two	205.5	0.06	67.74
$1.78 \cdot 10^6$	256	0.03	two	209.5	0.12	71.16
$3.16 \cdot 10^6$	512	0.0	one	882.7	0.00	78.27
$3.16 \cdot 10^6$	512	0.004	two	$1.52 \cdot 10^3$	0.15	143.82
$3.16 \cdot 10^6$	512	0.02	two	325.6	0.15	66.38
$3.16 \cdot 10^6$	512	0.04	two	485.8	0.24	83.74
$7.09 \cdot 10^6$	1024	0.0	one	$5.55 \cdot 10^3$	0.00	426.07
$7.09 \cdot 10^6$	1024	0.02	two	522.3	0.15	131.13
$7.09 \cdot 10^6$	1024	0.03	two	$4.17 \cdot 10^3$	0.38	548.94

Table 4.2

Performance of two-level method for straight tube problem with various parameter values for the Eisenstat–Walker linear tolerances and coarse tolerances. $\eta_h^{r,0}$ *is the initial tolerance for the linear relative residual in the Eisenstat–Walker method, and "coarse time" is the fraction of total compute time spent on the coarse grid. For* 512 *processors we have* $3.16 \cdot 10^6$ *unknowns, while for* 1024 *we have* $7.09 \cdot 10^6$ *unknowns.*

np	$\eta_h^{r,0}$	η_H^r	Coarse time	fGMRES	Walltime
512	10^{-3}	10^{-3}	0.14	335.1	71.01
512	10^{-3}	10^{-4}	0.19	332.9	75.19
512	10^{-3}	10^{-6}	0.29	333.9	85.31
512	10^{-4}	10^{-3}	0.15	326.6	66.81
512	10^{-4}	10^{-4}	0.2	328.0	70.64
512	10^{-4}	10^{-6}	0.3	329.3	79.97
512	10^{-6}	10^{-3}	0.16	387.3	70.73
512	10^{-6}	10^{-4}	0.22	379.2	74.8
512	10^{-6}	10^{-6}	0.33	377.2	85.36
1024	10^{-3}	10^{-3}	0.14	471.1	126.72
1024	10^{-3}	10^{-4}	0.2	469.6	136.68
1024	10^{-3}	10^{-6}	0.32	472.6	158.28
1024	10^{-4}	10^{-3}	0.16	522.3	130.29
1024	10^{-4}	10^{-4}	0.22	515.3	138.26
1024	10^{-4}	10^{-6}	0.35	517.2	161.42
1024	10^{-6}	10^{-3}	0.16	555.0	134.88
1024	10^{-6}	10^{-4}	0.23	554.3	145.36
1024	10^{-6}	10^{-6}	0.35	559.1	172.4

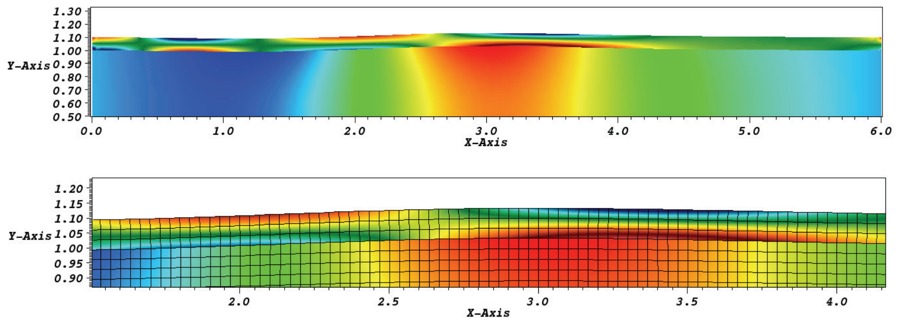

Fig. 4.4. *A plot of fluid and structure pressure for the straight tube case, with the top half of the tube visible in the top figure (the solution is vertically symmetric) and a zoomed-in portion of the same solution below, with the mesh outlined in black. For this smooth geometry and smooth deformation, we get a good result with a fairly coarse mesh.*

4.5. Here we use a Young's modulus of $E_s = 3.0 \cdot 10^4$ kg/m s^2, and the structure is again incompressible and has a density of 1000 kg/m^3. We start the simulation from rest, with an impulsive Dirichlet inlet velocity condition of 0.05 m/s, and we restart flexible GMRES every 50 iterations.

In this more physically realistic and computationally challenging example using a branching artery geometry, the difference in linear iteration counts between one- and two-level methods is even more marked. In Table 4.3, which shows average results for

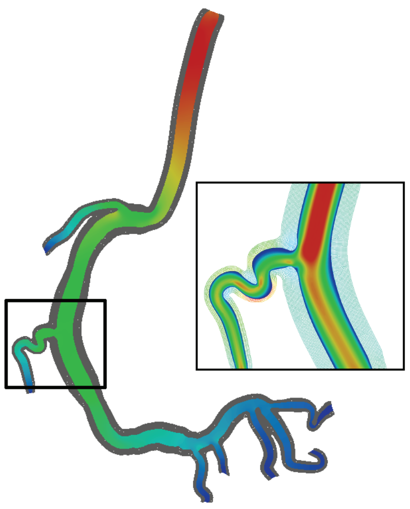

FIG. 4.5. *Results of a simulation done on a realistic, clinically acquired geometry. In the larger image the fluid is shaded by pressure and the structure is a solid shade, and in the inset image the fluid is shaded by norm of velocity and the structure mesh is shown shaded by norm of displacement.*

several computations, the two-level method results in a very sharp reduction in linear iterations and a good reduction in compute time for these problems and is much more effective than the one-level preconditioner. The two-level method is more robust to a variety of physical parameters. In particular, in Figure 4.6, we show the dramatic difference in total number of linear iterations per timestep between the one-level and two-level methods for varying timestep sizes. As the timestep size Δt is increased, the number of iterations for the one-level method increases dramatically, while those for the two-level method are fairly stable.

The overlap parameter δ in the Schwarz domain decomposition method is one way to adjust the strength of the preconditioner—a higher δ means more information transfer between subdomains, and therefore a faster convergence. This information transfer is expensive and results in larger local problems, so of course there is a tradeoff. Another way to exchange information between subdomains is with a coarse grid, and in Tables 4.4 and 4.5 it is clear that in the two-level method, the need to use overlap is greatly reduced. It also seems to be the case that the two-level method is less sensitive to the choice of overlap parameter δ.

TABLE 4.3

*Solver characteristics for increasing number of subdomains, with fixed problem size (*1.63 *million unknowns) and fixed overlap parameter (*$\delta = 0$ *for the two-level method,* $\delta = 3$ *for the one-level method), with comparisons of fGMRES iterations and walltime per timestep for the one-level method against the two-level method.*

	fGMRES iterations		Walltime	
Subdomains	One-level	Two-level	One-level	Two-level
96	442	237	270	184
112	514	245	277	182
128	487	286	216	163
160	697	282	201	105
192	899	485	168	109
224	1040	349	152	91.1
256	1020	382	127	79.9

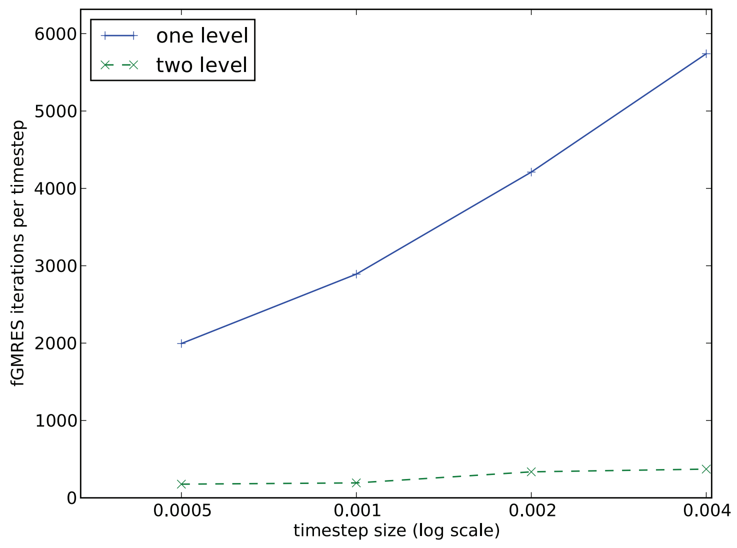

Fig. 4.6. *Linear iterations per timestep for various sizes of* Δt*, with comparison between the one-level and two-level methods. This is for a grid with* $2.40 \cdot 10^6$ *unknowns on* 256 *processors.*

One difficulty in the simulation of blood flow under realistic conditions is that, for an incompressible structure, as the Young's modulus E_s increases, the difficulty of the linear Jacobian solves and the number of linear iterations increase correspondingly. This difficulty, in fact, was one of the primary motivators for adding a two-level preconditioner to the method proposed in [5]. That difficulty remains for the two-level method, but because the two-level linear solver performs so much better, we are able to perform simulations more easily for more realistic values of E_s. See Table 4.6, where the two-level solver performs fairly well for a wide range of E_s values.

One drawback of the proposed two-level method is that we have a very large number of solver parameters to consider, all of which may have some effect on the performance of the method. These include the linear solver tolerances on the fine

Table 4.4

Overlap parameter comparisons for one-level and two-level methods on a branching grid. Because of the global exchange of information that the coarse grid provides, the optimal overlap is lower in the two-level case than in the one-level case. Also note that the one-level case fails to converge at all in the case of zero overlap, while the two-level method performs well. Each case above has $1.63 \cdot 10^6$ *unknowns.*

		One-level method		Two-level method	
np	δ	fGMRES	Walltime	fGMRES	Walltime
128	0	—	—	241.2	137.86
	1	$2.35 \cdot 10^3$	406.32	261.4	186.48
	2	820.6	270.19	225.2	210.11
	3	487.4	214.43	201.4	193.15
	4	356.6	225.61	180.2	210.68
256	0	—	—	360.4	59.24
	1	$3.84 \cdot 10^3$	324.88	417.2	83.18
	2	$2.39 \cdot 10^3$	254.42	376.0	129.83
	3	$1.02 \cdot 10^3$	127.86	329.4	95.16
	4	696.2	137.46	295.6	123.36

Table 4.5

More overlap parameter comparisons for one-level and two-level methods on a branching grid—see Table 4.4. Each case below has $2.40 \cdot 10^6$ unknowns.

		One-level method		Two-level method	
np	δ	fGMRES	Walltime	fGMRES	Walltime
256	0	—	—	423.2	114.98
	1	—	—	413.4	135.23
	2	$3.16 \cdot 10^3$	340.23	338.2	148.22
	4	$1.02 \cdot 10^3$	207.86	433.2	194.31
512	0	—	—	$1.10 \cdot 10^3$	108.04
	1	—	—	879.0	108.26
	2	—	—	$1.17 \cdot 10^3$	128.1
	4	$5.94 \cdot 10^3$	471.48	737.0	121.41

Table 4.6

Comparison of the methods for various values of the structure Young's modulus E_s, which can be thought of as representing the stiffness of the structure. A stiffer structure is more difficult to represent in terms of linear iterations, but the two-level method performs well for a wide range of μ_s. The heading "coarse time" is the fraction of total compute time spent on the coarse grid.

Unknowns	np	Type	E_s	fGMRES	Coarse time	Walltime
$1.63 \cdot 10^6$	128	two	$7.50 \cdot 10^3$	143.0	0.04	403.2
	128	two	$7.50 \cdot 10^4$	322.0	0.12	292.72
	128	two	$7.50 \cdot 10^5$	816.0	0.37	410.74
	128	one	$7.50 \cdot 10^3$	314.5	0.0	325.05
	128	one	$7.50 \cdot 10^4$	$1.57 \cdot 10^3$	0.0	418.73
$1.63 \cdot 10^6$	256	two	$7.50 \cdot 10^3$	229.5	0.1	145.55
	256	two	$7.50 \cdot 10^4$	487.0	0.27	125.13
	256	two	$7.50 \cdot 10^5$	$1.03 \cdot 10^3$	0.68	203.63
	256	one	$7.50 \cdot 10^3$	525.0	0.0	120.26
	256	one	$7.50 \cdot 10^4$	$3.53 \cdot 10^3$	0.0	273.12
$2.40 \cdot 10^6$	128	two	$7.50 \cdot 10^3$	214.0	0.04	679.21
	128	two	$7.50 \cdot 10^4$	445.0	0.1	509.28
	128	two	$7.50 \cdot 10^5$	$1.05 \cdot 10^3$	0.31	693.27
	128	one	$7.50 \cdot 10^3$	449.0	0.0	561.32
	128	one	$7.50 \cdot 10^4$	$2.27 \cdot 10^3$	0.0	788.01
$2.40 \cdot 10^6$	256	two	$7.50 \cdot 10^3$	276.0	0.08	212.36
	256	two	$7.50 \cdot 10^4$	627.5	0.23	187.55
	256	two	$7.50 \cdot 10^5$	$1.39 \cdot 10^3$	0.56	306.3
	256	one	$7.50 \cdot 10^3$	710.0	0.0	191.19
	256	one	$7.50 \cdot 10^4$	$4.38 \cdot 10^3$	0.0	445.22

and coarse levels, the nonlinear tolerances on the coarse level, the fGMRES restart parameters, the overlap parameters, the discretization size on the coarse level, and whether we choose left or right preconditioning. The choices of these parameters can have strong effects on the efficiency and parallel scaling of the method, and the different parameters can interact with each other in complex and difficult-to-predict ways. As an example, we present in Tables 4.2 and 4.7 some results for varying linear solver tolerances for the two-level method. We choose our parameters based on many such results, but these are difficult and sometimes expensive to obtain and cannot always give the best parameter choice for a given situation. A more systematic way of selecting parameters, based on some predictive theory, would be a useful avenue for future research.

4.3. Two-level Newton results. Our primary motivation for introducing a coarse mesh and associated interpolation and restriction operators was to reduce the

Table 4.7
Comparison of different stopping criteria for the two-level method on a branching grid. The entries marked with asterisks indicate the best choice of parameters for reducing total compute time.

np	Unknowns	η_h^r	η_H^r	Newton	fGMRES	Walltime
128	$8.33 \cdot 10^5$	10^{-4}	10^{-4}	3.6	243.0	142.49
128*	$8.33 \cdot 10^5$	10^{-4}	10^{-3}	3.6	323.4	124.87*
128	$8.33 \cdot 10^5$	10^{-3}	10^{-4}	3.6	190.2	125.74
128	$8.33 \cdot 10^5$	10^{-3}	10^{-3}	3.6	266.8	113.65
128	$8.33 \cdot 10^5$	10^{-6}	10^{-4}	3.6	369.6	191.8
128	$8.33 \cdot 10^5$	10^{-6}	10^{-3}	3.6	480.4	161.36
128	$1.63 \cdot 10^6$	10^{-4}	10^{-4}	3.0	346.8	312.56
128*	$1.63 \cdot 10^6$	10^{-4}	10^{-3}	3.0	490.4	289.58*
128	$1.63 \cdot 10^6$	10^{-3}	10^{-4}	3.4	293.8	320.84
128	$1.63 \cdot 10^6$	10^{-3}	10^{-3}	3.4	436.2	308.15
128	$1.63 \cdot 10^6$	10^{-6}	10^{-4}	3.0	487.2	373.47
128	$1.63 \cdot 10^6$	10^{-6}	10^{-3}	3.0	658.4	338.4
128*	$2.40 \cdot 10^6$	10^{-4}	10^{-4}	3.0	428.0	359.5*
128	$2.40 \cdot 10^6$	10^{-3}	10^{-4}	3.6	408.6	400.91
128	$2.40 \cdot 10^6$	10^{-3}	10^{-3}	3.6	552.4	377.43
128	$2.40 \cdot 10^6$	10^{-6}	10^{-4}	3.0	626.6	455.05
128	$2.40 \cdot 10^6$	10^{-6}	10^{-3}	3.0	791.6	407.15
256	$8.33 \cdot 10^5$	10^{-4}	10^{-4}	3.0	310.8	90.94
256*	$8.33 \cdot 10^5$	10^{-4}	10^{-3}	3.0	484.8	71.41*
256	$8.33 \cdot 10^5$	10^{-3}	10^{-4}	3.8	292.8	96.27
256	$8.33 \cdot 10^5$	10^{-3}	10^{-3}	3.8	467.0	79.31
256	$8.33 \cdot 10^5$	10^{-6}	10^{-4}	3.0	436.0	119.81
256	$8.33 \cdot 10^5$	10^{-6}	10^{-3}	3.0	647.2	91.93
256	$1.63 \cdot 10^6$	10^{-4}	10^{-4}	3.0	452.2	220.51
256*	$1.63 \cdot 10^6$	10^{-4}	10^{-3}	3.0	640.6	191.32*
256	$1.63 \cdot 10^6$	10^{-3}	10^{-4}	3.6	429.2	238.53
256	$1.63 \cdot 10^6$	10^{-3}	10^{-3}	3.6	573.0	206.99
256	$1.63 \cdot 10^6$	10^{-6}	10^{-4}	3.0	660.0	290.27
256	$1.63 \cdot 10^6$	10^{-6}	10^{-3}	3.0	834.0	229.17
256	$2.40 \cdot 10^6$	10^{-4}	10^{-4}	3.6	742.0	330.55
256	$2.40 \cdot 10^6$	10^{-4}	10^{-3}	3.6	978.2	284.64
256	$2.40 \cdot 10^6$	10^{-3}	10^{-4}	3.6	553.4	270.07
256*	$2.40 \cdot 10^6$	10^{-3}	10^{-3}	3.6	760.2	241.45*
256	$2.40 \cdot 10^6$	10^{-6}	10^{-4}	3.6	$1.07 \cdot 10^3$	447.3
256	$2.40 \cdot 10^6$	10^{-6}	10^{-3}	3.6	$1.42 \cdot 10^3$	369.23

number of linear iterations for problems with a large number of subdomains by using a coarse solve as a preconditioner for the linear solver. But once the data structure and the interpolation are developed, it only makes sense to use the coarse mesh as much as possible, and one simple way to use it is to do a nonlinear solve on the coarse grid to provide a better initial guess for the Newton solve on the fine grid. In Table 4.8, we show the modest but certainly noticeable effect of this two-level Newton approach, even when used without any two-level linear preconditioning. Its effect in reducing the number of nonlinear and also, interestingly, linear iterations is visible. For practical computational purposes it is usually advantageous to use a very coarse coarse grid, to minimize computational cost, but there is some benefit in iteration counts for using a relatively fine coarse grid—see Table 4.9.

TABLE 4.8

Comparison of one-level algorithm to two-level Newton method with only a one-level linear solver—in both cases we use a one-level preconditioner for the linear solver. Modest improvements in nonlinear iterations counts, walltimes, and, interestingly, linear iterations, are seen from using a coarse nonlinear solver. The first two lines (with $4.51 \cdot 10^5$ *unknowns) are for a straight tube, while the other results are for the branching model geometry.*

Unknowns	np	One-level Newton	Newton fGMRES	Walltime	Two-level Newton	Newton fGMRES	Walltime
$4.51 \cdot 10^5$	256	3.8	153.1	10.59	3.4	117.8	10.17
$4.51 \cdot 10^5$	512	3.5	258.42	5.94	3.4	197.9	6.45
$2.40 \cdot 10^6$	128	3.0	2910	756.8	2.67	2570	722.4
$2.40 \cdot 10^6$	256	3.0	8970	686.3	2.67	7190	621.7
$3.67 \cdot 10^6$	256	3.0	12000	1530	3.0	9510	1320

TABLE 4.9

Effect of different coarse-grid sizes on the two-level Newton method. These results are for the straight tube test problem—"coarse size" is the number of unknowns on the coarse mesh as fraction of unknowns on the fine level, and "coarse time" is the fraction of total compute time spent on the coarse grid.

Unknowns	np	Coarse size	Levels	Newton	fGMRES	Coarse time	Walltime
$4.51 \cdot 10^5$	64	0.0	one	3.8	74.6	0.00	46.24
	64	0.03	two	3.7	64.2	0.02	45.98
	64	0.12	two	3.4	58.9	0.07	43.24
$7.97 \cdot 10^5$	128	0.0	one	3.6	123.2	0.00	41.58
	128	0.02	two	3.6	103.1	0.01	41.04
	128	0.07	two	3.4	97.9	0.05	40.12
	128	0.14	two	3.5	102.4	0.09	42.96
$1.78 \cdot 10^6$	256	0.0	one	3.4	313.5	0.00	65.22
	256	0.01	two	3.5	243.8	0.01	63.49
	256	0.03	two	3.5	247.2	0.02	64.51
	256	0.06	two	3.5	256.2	0.03	65.52

4.4. Different coarse-grid solves. In the above, we have used one-level additive Schwarz preconditioned GMRES as the solver on the coarse level. There are of course many other options for this solver—more so than on the fine grid, because the coarse grid can still be quite effective as a preconditioner even if the solver is not very accurate. It is especially useful to be able to control costs on the coarse grid, since the coarse solve becomes the scalability bottleneck—see Figure 4.3. Here we consider some other strategies—for a more thorough study using similar algorithms for a different application, see [28].

One possibility is to use a simpler solver, rather than GMRES, on the coarse level. Applying a few Richardson sweeps preconditioned with one-level RAS (restricted additive Schwarz) to get a rough coarse-grid correction rather than solving to a specified tolerance with GMRES may save some computational effort, since Richardson sweeps are cheaper than GMRES. On the other hand, we can set the residual tolerance for GMRES in order to control costs. In Table 4.10, you can see the effect of a fixed number of Richardson iterations on the coarse level compared to our standard GMRES method—though the Richardson methods show some potential for improving the algorithm, especially for large problems, the GMRES method also performs well and is easier to implement.

TABLE 4.10
Comparison of different coarse-grid solvers. Though fine-tuning the exact number of Richardson iterations might be cost-effective in some cases, our choice of GMRES is robust and compares well with the Richardson options. The heading "coarse time" is the fraction of total compute time spent on the coarse grid.

Unknowns	np	Type	fGMRES	Coarse time	Walltime
$4.51 \cdot 10^5$	64	GMRES	73.5	0.04	42.82
	64	Rich(1)	89.0	0.01	45.09
	64	Rich(3)	73.2	0.04	43.03
	64	Rich(10)	72.3	0.21	50.4
$7.97 \cdot 10^5$	128	GMRES	116.2	0.05	36.91
	128	Rich(1)	130.2	0.02	36.71
	128	Rich(3)	116.2	0.04	36.99
	128	Rich(10)	115.5	0.23	44.66
$1.78 \cdot 10^6$	256	GMRES	242.2	0.07	59.91
	256	Rich(1)	285.0	0.03	60.99
	256	Rich(3)	255.2	0.04	58.9
	256	Rich(10)	241.9	0.06	59.55
$3.16 \cdot 10^6$	512	GMRES	441.2	0.13	63.04
	512	Rich(1)	707.0	0.06	73.22
	512	Rich(3)	478.3	0.06	62.51
	512	Rich(10)	441.5	0.10	63.76

5. Conclusion.

In this paper we have developed and studied two-level Newton–Krylov–Schwarz methods for fluid-structure interaction, which is a difficult and important problem. In this framework there are a large number of choices and interacting solvers, which we have examined. We have demonstrated effective, scalable preconditioners for the fully coupled monolithic problem that allow us to efficiently simulate blood flow on complicated realistic geometries and within realistic parameter regimes.

REFERENCES

[1] S. Badia, A. Quaini, and A. Quarteroni, *Modular vs. non-modular preconditioners for fluid-structure systems with large added-mass effect*, Comput. Methods Appl. Mech. Engrg., 197 (2008), pp. 4216–4232.

[2] S. Badia, A. Quaini, and A. Quarteroni, *Splitting methods based on algebraic factorization for fluid-structure interaction*, SIAM J. Sci. Comput., 30 (2008), pp. 1778–1805.

[3] S. Balay, K. Buschelman, V. Eijkhout, W. D. Gropp, D. Kaushik, M. G. Knepley, L. C. McInnes, B. F. Smith, and H. Zhang, *PETSc Users Manual*, Technical report, Argonne National Laboratory, Argonne, IL, 2008.

[4] A. T. Barker, *Monolithic Fluid-Structure Interaction Algorithms for Parallel Computing with Application to Blood Flow*, Ph.D. thesis, University of Colorado, Boulder, 2009.

[5] A. T. Barker and X.-C. Cai, *Scalable parallel methods for monolithic coupling in fluid-structure interaction with application to blood flow modeling*, J. Comput. Phys., 229 (2010), pp. 642–659.

[6] X.-C. Cai, *Additive Schwarz algorithms for parabolic convection-diffusion equations*, Numer. Math., 60 (1990), pp. 42–62.

[7] X.-C. Cai and M. Sarkis, *A restricted additive Schwarz preconditioner for general sparse linear systems*, SIAM J. Sci. Comput., 21 (1999), pp. 792–797.

[8] P. Causin, J. F. Gerbeau, and F. Nobile, *Added-mass effect in the design of partitioned algorithms for fluid-structure problems*, Comput. Methods Appl. Mech. Engrg., 194 (2005), pp. 4506–4527.

[9] T. F. Chan, B. F. Smith, and J. Zou, *Overlapping Schwarz methods on unstructured meshes using non-matching coarse grids*, Numer. Math., 73 (1996), pp. 149–167.

[10] C. R. Dohrmann and O. B. Widlund, *An overlapping Schwarz algorithm for almost incompressible elasticity*, SIAM J. Numer. Anal., 47 (2009), pp. 2897–2923.

[11] J. Donea, A. Huerta, J.-P. Ponthot, and A. Rodríguez-Ferran, *Arbitrary Lagrangian-Eulerian methods*, in Encyclopedia of Computational Mechanics, Vol. 1, E. Stein, R. de Borst, and T. J. Hughes, eds., Wiley, New York, 2004, pp. 1–25.

[12] E. Efstathiou and M. J. Gander, *Why restricted additive Schwarz converges faster than additive Schwarz*, BIT, 43 (2003), pp. 945–959.

[13] S. C. Eisenstat and H. F. Walker, *Choosing the forcing terms in an inexact Newton method*, SIAM J. Sci. Comput., 17 (1996), pp. 16–32.

[14] C. Farhat and P. Geuzaine, *Design and analysis of robust ALE time-integrators for the solution of unsteady flow problems on moving grids*, Comput. Methods Appl. Mech. Engrg., 193 (2004), pp. 4073–4095.

[15] L. Fatone, P. Gervasio, and A. Quarteroni, *Multimodels for incompressible flows*, J. Math. Fluid Mech., 2 (2000), pp. 126–150.

[16] C. A. Figueroa, I. E. Vignon-Clementel, K. E. Jansen, T. J. R. Hughes, and C. A. Taylor, *A coupled momentum method for modeling blood flow in three-dimensional deformable arteries*, Comput. Methods Appl. Mech. Engrg., 195 (2006), pp. 5685–5706.

[17] P. F. Fischer, *An overlapping Schwarz method for spectral element solution of the incompressible Navier-Stokes equations*, J. Comput. Phys., (1997), pp. 84–101.

[18] L. Grinberg and G. E. Karniadakis, *A scalable domain decomposition method for ultra-parallel arterial flow simulations*, Commun. Comput. Phys., 4 (2008), pp. 1151–1169.

[19] D. Hawken, J. Gottlieb, and J. Hansen, *Review of some adaptive node-movement techniques in finite-element and finite-difference solutions of partial differential equations*, J. Comput. Phys., 95 (1991), pp. 254–302.

[20] G. A. Holzapfel, T. C. Gasser, and R. W. Ogden, *A new constitutive framework for arterial wall mechanics and a comparative study of material models*, J. Elasticity, 61 (2000), pp. 1–48 (2001), pp. 1–48.

[21] B. Hübner, E. Walhron, and D. Dinkler, *A monolithic approach to fluid-structure interaction using space-time finite elements*, Comput. Methods Appl. Mech. Engrg., 193 (2004), pp. 2087–2104.

[22] F.-N. Hwang and X.-C. Cai, *Parallel fully coupled Schwarz preconditioners for saddle-point problems*, Electron. Trans. Numer. Anal., 22 (2006), pp. 146–162.

[23] G. Karypis, *Metis/Parmetis Web Page*, University of Minnesota, 2008, http://glaros.dtc.umn.edu/gkhome/views/metis.

[24] A. Klawonn and L. F. Pavarino, *Overlapping Schwarz methods for mixed linear elasticity and Stokes problems*, Comput. Methods Appl. Mech. Engrg., 165 (1998), pp. 233–245.

[25] L. Kness, *Parallel Fully-Coupled Fluid Structure Interaction Simulation of Several Benchmark Problems with Scalability Results*, M.S. thesis, University of Colorado, Boulder, 2009.

[26] J. Li, *A dual-primal FETI method for incompressible Stokes equations*, Numer. Math., 102 (2005), pp. 257–275.

[27] J. Li and O. Widlund, *BDDC algorithms for incompressible Stokes equations*, SIAM J. Numer. Anal., 44 (2006), pp. 2432–2455.

[28] P. T. Lin, J. N. Shadid, M. Sala, R. S. Tuminaro, G. L. Hennigan, and R. J. Hoekstra, *Performance of a parallel algebraic multilevel preconditioner for stabilized finite element semiconductor device modeling*, J. Comput. Phys., 228 (2009), pp. 6250–6267.

[29] X. Ma, G. Lee, and S. Wu, *Numerical simulation for the propagation of nonlinear pulsatile waves in arteries*, Trans. ASME, 114 (1992), pp. 490–496.

[30] L. F. Pavarino, *Indefinite overlapping Schwarz methods for time-dependent Stokes problems*, Comput. Methods Appl. Mech. Engrg., 187 (2000), pp. 35–51.

[31] L. F. Pavarino and S. Scacchi, *Multilevel additive Schwarz preconditioners for the bidomain reaction-diffusion system*, SIAM J. Sci. Comput., 31 (2008), pp. 420–443.

[32] Y. Saad, *Iterative Methods for Sparse Linear Systems*, 2nd ed., SIAM, Philadelphia, 2003.

[33] S. Scacchi, *A hybrid multilevel Schwarz method for the bidomain model*, Comput. Methods Appl. Mech. Engrg., 197 (2008), pp. 4051–4061.

[34] B. Smith, P. Bjørstad, and W. Gropp, *Domain Decomposition*, Cambridge University Press, Cambridge, UK, 1996.

[35] K. Stüben, *A review of algebraic multigrid*, J. Comput. Appl. Math., 128 (2001), pp. 281–309.

[36] C. A. Taylor and M. T. Draney, *Experimental and computational methods in cardiovascular fluid mechanics*, Ann. Rev. Fluid Mech., 36 (2004), pp. 197–231.

[37] A. Toselli and O. Widlund, *Domain Decomposition Methods—Algorithms and Theory*, Springer-Verlag, Berlin, 2005.

[38] A. M. Winslow, *Adaptive-Mesh Zoning by the Equipotential Method*, Technical report, Argonne National Laboratory, Argonne, IL, 1981.