MULTISCALE MODEL. SIMUL.
Vol. 10, No. 3, pp. 853–873

A FROZEN JACOBIAN MULTISCALE MORTAR PRECONDITIONER FOR NONLINEAR INTERFACE OPERATORS*

BENJAMIN GANIS†‡, GERGINA PENCHEVA†, MARY F. WHEELER†, TIM WILDEY§, AND IVAN YOTOV¶

Abstract. We present an efficient approach for preconditioning systems arising in multiphase flow in a parallel domain decomposition framework known as the mortar mixed finite element method. Subdomains are coupled together with appropriate interface conditions using mortar finite elements. These conditions are enforced using an inexact Newton–Krylov method, which traditionally required the solution of nonlinear subdomain problems on each interface iteration. A new preconditioner is formed by constructing a multiscale basis on each subdomain for a fixed Jacobian and time step. This basis contains the solutions of nonlinear subdomain problems for each degree of freedom in the mortar space and is applied using an efficient linear combination. Numerical experiments demonstrate the relative computational savings of recomputing the multiscale preconditioner sparingly throughout the simulation versus the traditional approach.

Key words. multiscale, mortar finite element, domain decomposition, multiphase flow, nonlinear interface problem

AMS subject classifications. 65M55, 65M60, 76S05

DOI. 10.1137/110826643

1. Introduction.

Driven by important applications in energy and the environment, the accurate simulation of realistic flow through the subsurface involves highly heterogeneous media on large domains. Even under basic modeling assumptions, these systems are computationally intensive and require specialized techniques for efficient solution. Nonoverlapping domain decomposition methods are a popular approach that allows parallel implementation, physically meaningful interface conditions, and the possibility of coupling multiple physical and numerical models.

In this work we consider the multiscale mortar mixed finite element method (MMMFEM) applied to multiphase flow. This type of method was first introduced for elliptic problems on nonmatching grids in [6], and the analysis was extended to the multiscale case in [7]. It serves as an alternative to other multiscale methods, such as the variational multiscale method [19, 20, 4, 1, 5, 3] and multiscale finite elements [17, 18, 14, 12, 21, 2], which are closely related [5]. In all three methods,

*Received by the editors March 7, 2011; accepted for publication (in revised form) April 26, 2012; published electronically August 2, 2012. This material is based upon work supported as part of the Center for Frontiers of Subsurface Energy Security, an Energy Frontier Research Center funded by the U.S. Department of Energy, Office of Science, Office of Basic Energy Sciences under award DE-SC0001114. The first and fifth authors were partially supported by the DOE grant DE-FG02-04ER25618 and the NSF grant DMS 0813901.

http://www.siam.org/journals/mms/10-3/82664.html

†Center for Subsurface Modeling, The Institute for Computational Engineering and Sciences (ICES), The University of Texas at Austin, Austin, TX 78712 (bganis@ices.utexas.edu, gergina@ices.utexas.edu, mfw@ticam.utexas.edu).

‡Corresponding author.

§Optimization and Uncertainty Quantification Department, Sandia National Laboratory, Albuquerque, NM 87185 (tmwilde@sandia.gov). Sandia National Laboratories is a multiprogram laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

¶Department of Mathematics, University of Pittsburgh, Pittsburgh, PA 15260 (yotov@math.pitt.edu).

the domain is decomposed into a series of small subdomains (coarse grid), and the solution is resolved globally on the coarse grid and locally (on each coarse element) on a fine grid. All three methods are based on a divide-and-conquer approach: solving relatively small fine scale subdomain problems that are only coupled together through a reduced number (coarse scale) of degrees of freedom.

In the MMMFEM, coarse scale mortar finite elements are used to couple subdomain grids which may be nonmatching. Algorithmically, the unknowns are consolidated on the interfaces, and the system is solved with an iterative method. In this way, the interface operator need not be formed since only its action is needed; each interface iteration requires the solution to subdomain problems which are performed in parallel.

An alternative algorithm called the multiscale basis implementation was recently proposed in [15]. Here, a Dirichlet subdomain problem is performed for each mortar degree of freedom, and the resulting boundary fluxes are stored. When the model problem is linear, the solution to a subdomain problem may be computed with a linear combination of these multiscale basis functions. It was shown that in many cases this algorithm may perform better than a competitive balancing preconditioner [28]. Two extensions to this algorithm were proposed in [16] and [32] for stochastic models requiring an ensemble of realizations. In the former approach, a stochastic multiscale basis was formed that could be used across multiple realizations. In the latter approach, a deterministic multiscale basis was formed for a training operator associated with the mean permeability, and used as a very effective preconditioner. It was shown to be optimal in the sense that the condition number is independent of the number of subdomains as well as the subdomain and mortar discretizations.

The flow models used in this work have the additional complexity of incorporating fluid compressibility, multiple phases, gravity, and capillarity, leading to a nonlinear system of equations. When applying the MMMFEM, a Newton–Krylov method is used to solve both the subdomain problems and the interface problem. Once again the interface problem is solved with a matrix-free approach, where the action of the Jacobian is calculated using a forward difference approximation, as described in [34] and section 4.1.

The goal of this work is to extend the multiscale basis implementation to the (deterministic) nonlinear interface problem for compressible and multiphase flow. The fact that the model problem is nonlinear eliminates the notion that subdomain problems can be expressed as a superposition of multiscale basis functions. However, in this work we modify the multiscale basis implementation to construct a new type of interface preconditioner for linearized interface operators.

Similar to the training operator approach in [32], a multiscale basis is formed on each subdomain in parallel for the Jacobian associated with one interface Newton iteration and is then reused on subsequent Newton iterations. It is for this reason that we refer to this approach as a frozen Jacobian multiscale preconditioner. Since these problems are also time-dependent, it can be used over several time steps, or even throughout the entire simulation. We note that the formation of a multiscale basis can be costly, and therefore it should be computed sparingly.

We present numerical experiments that illustrate the efficiency of the preconditioner for single-phase and two-phase flow in porous media. The results indicate that the preconditioner significantly reduces the number of interface iterations. Furthermore, its effectiveness increases with the number of subdomains and the number of fine grid elements.

This paper is organized as follows. In section 2, the slightly compressible single-phase and two-phase flow models are introduced in the domain decomposition framework. In section 3, fully discrete schemes are formulated using the MMMFEM. In section 4, the problem is reduced to a nonlinear interface operator using an inexact Newton method. In section 5, we describe the frozen Jacobian multiscale preconditioner. Finally, in section 6 we present two numerical examples with a discussion of the results.

2. Flow models. We consider a computational domain, $\Omega \subset \mathbb{R}^d$, $d = 2$ or 3, decomposed into a series of nonoverlapping subdomains Ω_i, $i = 1, 2, \ldots, P$, with interfaces denoted by $\Gamma_{ij} = \partial\Omega_i \cap \partial\Omega_j$, $\Gamma_i = \bigcup_{j=1}^{P} \Gamma_{ij}$, and $\Gamma = \bigcup_{1 \le i < j \le P} \Gamma_{ij}$. A different flow model may be used on each subdomain if certain conditions are met along the interface [34]. For simplicity we assume that for a particular simulation the flow model is the same for each subdomain. The algorithms developed in this paper will be extended in a future work, by coupling single- with two- or three-phase models, similar to the approach in [29].

2.1. Single-phase model. If only a slightly compressible water phase is present, then the subsurface flow is characterized by the conservation-of-mass equation,

$$\frac{\partial}{\partial t}(\phi\rho) + \nabla \cdot \boldsymbol{u} = q, \tag{1}$$

where ϕ is the porosity, q is the source term, and $\boldsymbol{u}$ is the Darcy velocity given by

$$\boldsymbol{u} = -\frac{K}{\mu}\rho\left(\nabla p - \rho \boldsymbol{g}\right). \tag{2}$$

Here K is the permeability tensor, p is the pressure, μ is the viscosity, and $\boldsymbol{g}$ is the gravitational acceleration vector. The density, $\rho = \rho(p)$, satisfies the equation of state,

$$\rho = \rho^{\mathrm{ref}} e^{c(p - p^{\mathrm{ref}})}, \tag{3}$$

where ρ^{ref} is the reference density, p^{ref} is the reference pressure, and c is the compressibility. The nonlinearity of this model is rather modest since the compressibility is usually small. For simplicity, we assume no-flow boundary conditions $\boldsymbol{u} \cdot \boldsymbol{n} = 0$ on $\partial\Omega$, where $\boldsymbol{n}$ is the outward unit normal, and note that the extension to more general boundary conditions is straightforward.

2.2. Two-phase model. In the case of two-phase flow, we let the lowercase scripts w and o denote the water and oil phases, respectively. The corresponding phase saturations are denoted by S_w and S_o, the phase pressures by p_w and p_o, and the well injection/production rates by q_w and q_o.

We consider the two-phase immiscible slightly compressible oil-water flow model in which the densities of oil and water are given by the equation of state,

$$\rho_\alpha = \rho_\alpha^{\mathrm{ref}} e^{c_\alpha (p_\alpha - p_\alpha^{\mathrm{ref}})}, \tag{4}$$

where $\rho_\alpha^{\mathrm{ref}}$ is the reference density, p_α^{ref} is the reference pressure, and c_α is the compressibility for $\alpha = w, o$. The mass conservation equation and Darcy's law are

$$\frac{\partial}{\partial t}(\phi N_\alpha) + \nabla \cdot \boldsymbol{u}_\alpha = q_\alpha, \tag{5}$$

$$\boldsymbol{u}_\alpha = -\frac{k_\alpha K}{\mu_\alpha}\rho_\alpha\left(\nabla p - \rho_\alpha \boldsymbol{g}\right), \tag{6}$$

where $N_\alpha = S_\alpha \rho_\alpha$ is the phase concentration, $k_\alpha = k_\alpha(S_w)$ is the relative permeability, and μ_α is the density for $\alpha = w, o$. The above variables are subject to the constitutive constraints

$$S_o + S_w = 1, \quad p_c(S_w) = p_o - p_w.$$

The well injection and production rates are defined using the Peaceman well model [27] extended to multiphase and multicomponent flow, and they describe typical well conditions for pressure or rate specified wells. No-flow boundary conditions on $\partial\Omega$ are taken for each phase.

2.3. Interface conditions. To couple the subdomain models, we impose the physically meaningful interface conditions,

$$p_\alpha|_{\Omega_i} = p_\alpha|_{\Omega_j} \quad \text{on } \Gamma_{ij}, \tag{7}$$

$$[\boldsymbol{u}_\alpha \cdot \boldsymbol{n}]_{ij} := \boldsymbol{u}_\alpha|_{\Omega_i} \cdot \boldsymbol{n}_i + \boldsymbol{u}_\alpha|_{\Omega_j} \cdot \boldsymbol{n}_j = 0 \quad \text{on } \Gamma_{ij}, \tag{8}$$

where $\boldsymbol{n}_i$ denotes the unit outward normal on $\partial\Omega_i$. These conditions represent continuity of pressure and continuity of normal flux for each phase and are the basis for the domain decomposition algorithm to follow.

3. A fully discrete scheme. Let $(\cdot,\cdot)_{\Omega_i}$ denote the L^2 inner product over Ω_i, and $\langle\cdot,\cdot\rangle$ the usual duality pairing on Γ_{ij}. Let $\mathcal{T}_{h,i}$ be a conforming quasi-uniform affine finite element partition of Ω_i, $1 \le i \le P$, of maximal element diameter h_i. Note that we need quasi uniformity and conformity only on each subdomain. Our method allows for spatially varying h_i, but to simplify the discussion, we let $h = \max_{1\le i\le P} h_i$ and analyze the method in terms of this single value h. We allow for the possibility that $\mathcal{T}_{h,i}$ and $\mathcal{T}_{h,j}$ need not align on Γ_{ij}, and we define $\mathcal{T}_h = \cup_{i=1}^P \mathcal{T}_{h,i}$. Let

$$\boldsymbol{V}_{h,i} \times W_{h,i} \subset H(\mathrm{div};\Omega_i) \times L^2(\Omega_i)$$

be any of the usual mixed finite element spaces (e.g., those of [9, 10, 26, 11, 30]), and let $\boldsymbol{V}_h$ or, equivalently, $\boldsymbol{V}_h \cdot \boldsymbol{n}$ contain the polynomials of degree k. Then let

$$\boldsymbol{V}_h = \bigoplus_{i=1}^P \boldsymbol{V}_{h,i}, \quad W_h = \bigoplus_{i=1}^P W_{h,i}.$$

Note that the normal components of vectors in $\boldsymbol{V}_h$ are continuous between elements within each block Ω_i but not across Γ. We also define the subspaces

$$\boldsymbol{V}_{h,i,0} = \{\boldsymbol{v} \in \boldsymbol{V}_{h,i} \mid \boldsymbol{v}\cdot\boldsymbol{n}_i = 0 \text{ on } \partial\Omega_i \cap \partial\Omega\} \quad \text{and} \quad \boldsymbol{V}_{h,0} = \bigoplus_{i=1}^P \boldsymbol{V}_{h,i,0}.$$

Let the mortar interface mesh $\mathcal{T}_{H,ij}$ be a quasi-uniform finite element partition of Γ_{ij} with maximal element diameter H_{ij}. Let $H = \max_{1\le i,j\le P} H_{ij}$. Define $\mathcal{T}^{\Gamma,H} = \cup_{1\le i<j\le P}\mathcal{T}_{H,ij}$. Denote by $M_{H,ij} \subset L^2(\Gamma_{ij})$ the mortar space on Γ_{ij} containing either the continuous or discontinuous piecewise polynomials of degree m on $\mathcal{T}_{H,ij}$, where m is at least $k+1$ if nonmatching grids are used. In the case of matching grids we may

take $m = k$. We remark that $\mathcal{T}_{H,ij}$ need not be conforming if $M_{H,ij}$ is discontinuous. Now let

$$M_{H,i} = \bigoplus_{\substack{1\le j\le P,\\ \Gamma_{ij}\neq\emptyset}} M_{H,ij} \quad \text{and} \quad M_H = \bigoplus_{1\le i<j\le P} M_{H,ij}$$

be the mortar finite element spaces on Γ_i and Γ, respectively. For each subdomain Ω_i define a projection $\mathcal{Q}_{h,i} : L^2(\Gamma_i) \to \boldsymbol{V}_{h,i} \cdot \boldsymbol{n}_i|_{\Gamma_i}$ such that, for any $\varphi \in L^2(\Gamma_i)$,

$$\langle \varphi - \mathcal{Q}_{h,i}\varphi, \boldsymbol{v} \cdot \boldsymbol{n}_i \rangle_{\Gamma_i} = 0, \quad \boldsymbol{v} \in \boldsymbol{V}_{h,i}. \tag{9}$$

The only restriction on the mortar spaces is that for any $\eta \in M_{H,ij}$,

$$\mathcal{Q}_{h,i}\eta = \mathcal{Q}_{h,j}\eta = 0 \implies \eta = 0.$$

As noted in [6, 7, 33, 34], this condition is not very restrictive since the mortar space is usually much coarser than the trace of the subdomain grids.

Following [8, 34, 29], we use a variant of the mixed finite element method: the expanded method. This modified version allows for the proper treatment of the linear system in the degenerate case where one of the relative permeabilities is zero. For $\alpha = w, o$ let

$$\tilde{\boldsymbol{u}}_\alpha = -K(\nabla p_\alpha - \rho_\alpha \boldsymbol{g}).$$

Then

$$\boldsymbol{u}_\alpha = \lambda_\alpha \tilde{\boldsymbol{u}}_\alpha, \quad \text{where } \lambda_\alpha = \frac{k_\alpha \rho_\alpha}{\mu_\alpha}.$$

Define $0 = t_0 < t_1 < \cdots$, $\Delta t^n = t_n - t_{n-1}$, and $f^n = f(t_n)$ for any sufficiently smooth function f. The backward Euler expanded mortar mixed finite element method for the two-phase system (5)–(6) seeks $\tilde{\boldsymbol{u}}^n_{\alpha,h,i} \in \boldsymbol{V}_{h,i}$, $\boldsymbol{u}^n_{\alpha,h,i} \in \boldsymbol{V}_{h,i,0}$, $p^n_{\alpha,h,i} \in W_{h,i}$, $N^n_{\alpha,h,i} \in W_{h,i}$, and $p^n_{\alpha,H,i} \in M_{H,i}$ such that

$$\left(\phi \frac{N^n_{\alpha,h,i} - N^{n-1}_{\alpha,h,i}}{\Delta t^n}, w\right)_{\Omega_i} + \left(\nabla \cdot \boldsymbol{u}^n_{\alpha,h,i}, w\right)_{\Omega_i} = (q^n_\alpha, w)_{\Omega_i}, \qquad w \in W_{h,i}, \tag{10}$$

$$\left(K^{-1}\tilde{\boldsymbol{u}}^n_{\alpha,h,i}, \boldsymbol{v}\right)_{\Omega_i} = \left(p^n_{\alpha,h,i}, \nabla \cdot \boldsymbol{v}\right)_{\Omega_i} + (\rho_\alpha \boldsymbol{g}, \boldsymbol{v})_{\Omega_i} - \left\langle p^n_{\alpha,H,i}, \boldsymbol{v} \cdot \boldsymbol{n}_i \right\rangle_{\Gamma_i}, \qquad \boldsymbol{v} \in \boldsymbol{V}_{h,i,0}, \tag{11}$$

$$\left(\boldsymbol{u}^n_{\alpha,h,i}, \boldsymbol{z}\right)_{\Omega_i} = \left(\lambda_\alpha \tilde{\boldsymbol{u}}^n_{\alpha,h,i}, \boldsymbol{z}\right)_{\Omega_i}, \qquad \boldsymbol{z} \in \boldsymbol{V}_{h,i}. \tag{12}$$

This system is completed by the weak interface condition,

$$\sum_{1\le i<j\le P} \left\langle \left[\boldsymbol{u}^n_{\alpha,h} \cdot \boldsymbol{n}\right]_{ij}, \mu \right\rangle_{\Gamma_{ij}} = 0, \qquad \mu \in M_{H,ij}. \tag{13}$$

4. Reduction to a nonlinear interface operator. To simplify the notation we let

$$\boldsymbol{M}_H = \bigotimes_{\alpha} M_H$$

and define the nonlinear bivariate form $b^n : \boldsymbol{M}_H \times \boldsymbol{M}_H \to \mathbb{R}$ as

$$b^n(\boldsymbol{\varphi}, \boldsymbol{\mu}) := \sum_{i=1}^{P} \left\langle \boldsymbol{u}^n_{\alpha,h}(\boldsymbol{\varphi}) \cdot \boldsymbol{n}, \boldsymbol{\mu} \right\rangle_{\Gamma_i} = \sum_{1 \le i < j \le P} \left\langle \left[\boldsymbol{u}^n_{\alpha,h}(\boldsymbol{\varphi}) \cdot \boldsymbol{n} \right]_{ij}, \boldsymbol{\mu} \right\rangle_{\Gamma_{ij}}$$

for $\boldsymbol{\varphi}, \boldsymbol{\mu} \in \boldsymbol{M}_H$. For a given $\boldsymbol{\varphi}$, $[\boldsymbol{u}^n_{\alpha,h}(\boldsymbol{\varphi}) \cdot \boldsymbol{n}]$ is the jump in the normal flux obtained by solving nonlinear subdomain problems (10)–(12) with boundary data $p^n_{\alpha,H,i}(\boldsymbol{\varphi})$.

For single-phase flow, there is only one interface variable, and we take $\varphi = p^n_{w,H}$. On the other hand, for two-phase flow there are several choices for primary mortar variables. The most natural choice is $\boldsymbol{\varphi} = (p_{w,H}, p_{o,H})$. However, as discussed in [34], the interface operator will not be well defined if the problem becomes degenerate. A more suitable choice is to take $\boldsymbol{\varphi} = (p_{w,H}, N_{o,H})$ or $\boldsymbol{\varphi} = (p_{o,H}, N_{o,H})$. The missing pressure in each of these choices can be determined by the capillary pressure relationship, and the resulting interface form will be well defined. For more details on the choice of interface variables, see section 4.1 in [34]. For each subdomain i we define the nonlinear flux operator $B^n_i : \boldsymbol{M}_{H,i} \to \boldsymbol{M}_{H,i}$ by

$$\langle B^n_i \boldsymbol{\varphi}, \boldsymbol{\mu} \rangle_{\Gamma_i} = \left\langle \boldsymbol{u}^n_{\alpha,h}(\boldsymbol{\varphi}) \cdot \boldsymbol{n}_i, \boldsymbol{\mu} \right\rangle_{\Gamma_i},$$

and globally we define the nonlinear interface operator $B^n : \boldsymbol{M}_H \to \boldsymbol{M}_H$ by

$$\langle B^n \boldsymbol{\varphi}, \boldsymbol{\mu} \rangle_{\Gamma} = \sum_{i=1}^{P} \langle B^n_i \boldsymbol{\varphi}, \boldsymbol{\mu} \rangle_{\Gamma_i} = b^n(\boldsymbol{\varphi}, \boldsymbol{\mu}) \qquad \forall \boldsymbol{\mu} \in \boldsymbol{M}_H,$$

which measures the total flux jump. In [34] it is shown that if $\boldsymbol{\varphi}$ solves

$$B^n(\boldsymbol{\varphi}) = 0, \tag{14}$$

then $(p^n_{\alpha,h,i}, N^n_{\alpha,h,i}, \boldsymbol{u}^n_{\alpha,h,i}, \tilde{\boldsymbol{u}}^n_{\alpha,h,i})$ solves (10)–(13). We summarize the evaluation of the interface operator in Algorithm 1. Note that lines 1–4 compute the action of the subdomain flux operators B^n_i and can be performed in parallel, and line 5 computes the action of the interface operator B^n requiring interprocess communication.

ALGORITHM 1. Evaluation of the nonlinear interface operator.

1. Given interface data $\boldsymbol{\varphi}$.
2. Project $p^n_{\alpha,H,i}(\boldsymbol{\varphi})$ onto subdomain boundaries.
3. Solve independent nonlinear subdomain problems (10)–(12).
4. Project resulting flux $\boldsymbol{u}^n_{\alpha,h,i} \cdot \boldsymbol{n}_i$ onto the mortar space.
5. Sum fluxes over subdomains Ω_i to compute flux jump $B^n(\boldsymbol{\varphi})$.

4.1. An inexact Newton method. In the following we may suppress the time index superscript n. Since an analytic expression for the Jacobian of a nonlinear interface operator would require the exact expressions of the nonlinear Dirichlet-to-Neumann maps, we use an inexact Newton method to solve the nonlinear interface equation (14). The inexact Newton step $\boldsymbol{s}^{\{k\}} = \boldsymbol{\varphi}^{\{k+1\}} - \boldsymbol{\varphi}^{\{k\}}$ is computed by solving

$$D_\delta B(\boldsymbol{\varphi}^{\{k\}}; \boldsymbol{s}^{\{k\}}) = -B(\boldsymbol{\varphi}^{\{k\}}), \tag{15}$$

where the forward difference operator,

$$D_\delta B(\boldsymbol{\varphi};\boldsymbol{\mu}) = \begin{cases} \mathbf{0}, & \boldsymbol{\mu} = \mathbf{0}, \\ \|\boldsymbol{\mu}\| \frac{B(\boldsymbol{\varphi}+\delta\|\boldsymbol{\varphi}\|\boldsymbol{\mu}/\|\boldsymbol{\mu}\|)-B(\boldsymbol{\varphi})}{\delta\|\boldsymbol{\varphi}\|}, & \boldsymbol{\mu} \neq \mathbf{0}, \boldsymbol{\varphi} \neq \mathbf{0}, \\ \|\boldsymbol{\mu}\| \frac{B(\delta\boldsymbol{\mu}/\|\boldsymbol{\mu}\|)-B(\mathbf{0})}{\delta}, & \boldsymbol{\mu} \neq \mathbf{0}, \boldsymbol{\varphi} = \mathbf{0}, \end{cases} \tag{16}$$

is an approximation of the Jacobian $B'(\boldsymbol{\varphi})$ in the direction $\boldsymbol{\mu}$, following the approach used in [22] and [34]. Similarly, we can also define the local subdomain contributions $D_\delta B_i(\boldsymbol{\varphi};\boldsymbol{\mu})$ by replacing B with B_i in (16), and we note that $D_\delta B = \sum_i D_\delta B_i$.

The interface Newton update (15) is solved with a GMRES algorithm, where the action of the Jacobian is evaluated using (16) and the interface operator is evaluated using Algorithm 1. In most traditional implementations, only the action of the Jacobian is required, and the Jacobian itself is never constructed.

4.2. Preconditioning the Jacobian. A well-known drawback of iterative algorithms for interface operators is that the number of iterations, and therefore the number of subdomain solves, depends strongly on the subdomain and mortar discretizations, as well as the heterogeneities in the permeability. A good preconditioner is usually required to reduce the number of iterations to a reasonable level. A left-preconditioned GMRES strategy is based on solving

$$M^{-1} D_\delta B(\boldsymbol{\varphi}^{\{k\}}; \boldsymbol{s}^{\{k\}}) = -M^{-1} B(\boldsymbol{\varphi}^{\{k\}}), \tag{17}$$

where M is an easily invertible approximation to the Jacobian.

Physically, the interface operator $B(\boldsymbol{\varphi})$ is a Dirichlet-to-Neumann operator, while the preconditioner M represents a Neumann-to-Dirichlet operator. Thus, including a preconditioner in the GMRES algorithm means that consecutive Krylov vectors have the same physical interpretation.

The Neumann–Neumann [24] and balancing preconditioners [25, 28] are popular choices for linear interface operators due to their physical interpretation and ease of parallelization. To the best of our knowledge, the only preconditioner developed for the Jacobian of a nonlinear interface operator is the approximate Neumann–Neumann preconditioner described in [34]. Although this preconditioner is simple to apply, the reduction in the number of interface iterations is relatively small since it is essentially an approximate Jacobi preconditioner.

4.3. Construction of the interface Jacobian. We consider an alternative to the matrix-free method by constructing an approximation of the Jacobian based on the construction of a multiscale mortar flux basis [15, 32]. While the nonlinearity of the model problems eliminates the superposition property of multiscale basis functions, we use the multiscale basis to construct an approximation of the interface Jacobian. Let $\{\boldsymbol{\psi}_{H,i}^{\{m\}}\}$ for $m = 1, 2, \ldots, n_{H,i}$ denote a basis for the mortar space $\boldsymbol{M}_{H,i}$, where $n_{H,i}$ is the number of degrees of freedom associated with $\boldsymbol{M}_{H,i}$. For each $\boldsymbol{\psi}_{H,i}^{\{m\}}$, we compute a resulting flux $\boldsymbol{\chi}_{H,i}^{\{m\}}$ using Algorithm 2. We refer to $\{\boldsymbol{\chi}_{H,i}^{\{m\}}\}$ as the multiscale flux basis for subdomain Ω_i. Note that the multiscale basis functions depend on the chosen interface variables, $\boldsymbol{\varphi}^{\{k\}}$, and the Newton difference, δ. For each basis function, we then compute

$$\boldsymbol{\zeta}_{H,i}^{\{m\}} = \begin{cases} \|\boldsymbol{\psi}_{H,i}^{\{m\}}\| \frac{\boldsymbol{\chi}_{H,i}^{\{m\}} - B_i(\boldsymbol{\varphi})}{\delta\|\boldsymbol{\varphi}\|}, & \boldsymbol{\varphi} \neq \mathbf{0}, \\ \|\boldsymbol{\psi}_{H,i}^{\{m\}}\| \frac{\boldsymbol{\chi}_{H,i}^{\{m\}} - B_i(\mathbf{0})}{\delta}, & \boldsymbol{\varphi} = \mathbf{0}, \end{cases} \tag{18}$$

which we refer to as the multiscale Jacobian basis.

ALGORITHM 2. Construction of a multiscale flux basis $\{\boldsymbol{\chi}_{H,i}^{\{m\}}\}$.

for $i = 1, 2, \ldots, P$ **do**
  Given $\boldsymbol{\varphi}^{\{k\}}$ and δ.
  for $m = 1, 2, \ldots, n_{H,i}$ **do**
    Update the boundary data

$$p_{\alpha,H,i}^{n}\left(\boldsymbol{\varphi}^{\{k\}} + \delta\|\boldsymbol{\varphi}^{\{k\}}\|\boldsymbol{\psi}_{H,i}^{\{m\}}/\|\boldsymbol{\psi}_{H,i}^{\{m\}}\|\right).$$

    Solve the nonlinear subdomain problem (10)–(12).
    Project the resulting flux into the mortar space.
  end for
end for

The multiscale Jacobian functions are stored on the subdomain level on different processors, so that the global Jacobian need not be assembled. Instead, a Krylov method is used to solve for the Newton update in (15), which requires only the action of the Jacobian on each Krylov vector. This matrix-vector product is computed in parallel using the fact that each Krylov vector $\boldsymbol{v}_H$ is a linear combination of mortar basis functions, i.e.,

$$\boldsymbol{v}_H = \sum_{1=1}^{P} \boldsymbol{v}_{H,i} = \sum_{1=1}^{P}\sum_{m=1}^{n_{H,i}} c_i^{\{m\}}\boldsymbol{\psi}_{H,i}^{\{m\}}. \tag{19}$$

This allows us to compute

$$D_\delta B(\boldsymbol{\varphi}^{\{k\}};\boldsymbol{v}_H) \approx \sum_{i=1}^{P}\sum_{m=1}^{n_{H,i}} c_i^{\{m\}} D_\delta B_i(\boldsymbol{\varphi}^{\{k\}};\boldsymbol{\psi}_{H,i}^{\{m\}}) = \sum_{i=1}^{P}\sum_{m=1}^{n_{H,i}} c_i^{\{m\}}\boldsymbol{\zeta}_{H,i}^{\{m\}} \tag{20}$$

without assembling the global Jacobian. Note that, although $D_\delta B(\boldsymbol{\varphi}^{\{k\}};\boldsymbol{\mu})$ is in general nonlinear in $\boldsymbol{\mu}$, it is a good approximation to the linear operator $B'(\boldsymbol{\varphi}^{\{k\}})$ in the direction $\boldsymbol{\mu}$.

Since the Jacobian is a coarse scale operator, the cost in solving the problem is typically much smaller than the cost of solving subdomain problems. The cost associated with constructing the Jacobian directly depends on the number of degrees of freedom on each mortar, but only peripherally on the heterogeneities of the porous media. This is because the number of multiscale basis functions depends only on the mortar discretization, but computing each multiscale basis function requires subdomain solves, which are affected by heterogeneities. Therefore, this approach is favorable in the case of highly heterogeneous media with relatively few degrees of freedom in the mortar space and in the absence of a reasonable preconditioner. As mentioned in [7], a coarse mortar space can be used without sacrificing accuracy by taking higher order mortars if the problem is sufficiently regular.

5. A frozen Jacobian multiscale preconditioner. The two approaches for solving the linearized equation (15) for the Newton update, namely the matrix-free algorithm described in section 4.1 and the direct construction approach in section 4.3, have advantages and disadvantages depending on the size of the problem, the heterogeneities, and the availability of an adequate preconditioner. In Algorithm

ALGORITHM 3. Applying the frozen Jacobian multiscale preconditioner.

Given a multiscale Jacobian basis $M_i = [\boldsymbol{\zeta}_{H,i}^{\{m\}}]_{m=1}^{n_{H,i}}$ for each subdomain Ω_i associated with a fixed state $\hat{\boldsymbol{\varphi}}$.

Solve the preconditioned interface Newton step (17) using an outer GMRES iteration.

⋮

Given an outer Krylov vector $\boldsymbol{v}_H$.
Compute action $\boldsymbol{g}_H := D_\delta B(\boldsymbol{\varphi}^{\{k\}}; \boldsymbol{v}_H)$ by solving (10)–(12).
Solve $M\boldsymbol{w}_H = \boldsymbol{g}_H$ with an inner GMRES iteration.

⋮

Given an inner Krylov vector $\boldsymbol{y}_H$.
for $i = 1, \ldots, P$ **do**
Compute the linear combination $\boldsymbol{z}_{H,i} := M_i\ \boldsymbol{y}_{H,i}$, where $\boldsymbol{y}_{H,i} := \boldsymbol{y}_H|_{\Omega_i}$.
end for
Sum $\boldsymbol{z}_{H,i}$ over subdomains Ω_i to compute $M\boldsymbol{y}_H$.

⋮

end inner GMRES

⋮

end outer GMRES

3, we describe a way to combine these two approaches to solve the preconditioned Newton step (17).

The preconditioner M for the outer GMRES loop is not formed directly, so each application of M^{-1} requires an inner GMRES loop. Each inner GMRES iteration requires two steps. First, a linear combination is performed on the subdomain level with a matrix-vector product using M_i. Second, these products are summed across the subdomains using parallel communication. This procedure circumvents the need to solve any subdomain problems in the inner GMRES iteration. However, the parallel communication may lead to a nontrivial cost in runtime. In section 6, we see that this communication cost does indeed factor into the total simulation time, but the computational savings due to a reduced number of subdomain problems typically outweigh this cost.

Clearly, the performance of the preconditioner depends on how closely the Jacobian for $\hat{\boldsymbol{\varphi}}$ models the Jacobian for the current state. In many subsurface applications, the heterogeneity of the permeability field, which is captured by the initial Jacobian, dominates the characteristic features of the flow variables. Therefore, we expect the variance between the Jacobians to be relatively minor and to evolve slowly in time. The performance of the preconditioner can be monitored throughout the simulation, and a new preconditioner may be constructed based on the current state at any time.

A practical approach to determining at what time steps to recompute the multiscale preconditioner must effectively balance the overhead cost of constructing the current Jacobian using Algorithm 2 with its potential reduction in the number of interface iterations. We outline one such heuristic approach in the following section.

5.1. A heuristic for multiscale preconditioner recomputation. In both the slightly compressible single-phase and two-phase models, the subdomain problems are nonlinear and may vary in difficulty due to many factors related to the

current state of the system. However, let us assume that solving a typical subdomain problem has roughly the same computational cost. Under this assumption, a good heuristic will seek to minimize the number of subdomain problems that we must solve throughout the entire simulation. First recall that each interface GMRES iteration requires the solution to one subdomain problem, as well as an extra subdomain problem to form the right-hand side of each Newton update (15). We make the following definitions:

$$\begin{aligned}
N_{prob} &= \text{subdomain problems throughout entire simulation,}\\
N_{ms} &= \text{subdomain problems to recompute preconditioner,}\\
\mathcal{T} &= \{1,2,\ldots,N_{time}\}\text{, the total set of time indices,}\\
\mathcal{R} &= \{t_1,\ldots,t_k\}\text{, the set of } k \text{ recomputation time indices,}\\
\tilde{t}(n) &= \max\{r\in\mathcal{R}\mid r\le n\}\text{, the previous recomputation time,}\\
N_{unprec}(n) &= \text{unpreconditioned number of interface iterations at time } n,\\
N_{prec}(n,\tilde{t}(n)) &= \text{preconditioned number of interface iterations at time } n,\\
N_{base}(n,\tilde{t}(n)) &= \min\{N_{prec}(k,\tilde{t}(n))\mid k=\tilde{t}(n),\tilde{t}(n)+1,\ldots,n\},\\
N_{newt}(n) &= \text{number of Newton steps at time } n.
\end{aligned}$$

Each time we recompute the multiscale preconditioner, we must solve N_{ms} subdomain problems, equal to the number of mortar degrees of freedom times the number of phase variables. If we perform a simulation in parallel, then this number is local to each processor, determined by the number of subdomains which are "owned" by that processor.

When the preconditioner is applied to the interface problem at time n, it should dramatically reduce the number of interface iterations from N_{unprec} to N_{prec} for the current Jacobian. As the interface Jacobians change for subsequent Newton steps and time steps, the effectiveness of the preconditioner will degrade; i.e., N_{prec} increases as n gets further from $\tilde{t}(n)$. Therefore, there should be some baseline number of interface iterations, N_{base}, defined as the minimum number of interface iterations performed per time step since the last recomputation.

The problem is to find a set of recomputation times $\mathcal{R}\subset\mathcal{T}$ such that the number of total subdomain problems,

$$N_{prob} = \underbrace{kN_{ms}}_{recomputation} + \sum_{n=1}^{N_{time}} \left[N_{prec}(n,\tilde{t}(n)) + N_{newt}(n)\right], \tag{21}$$

is minimized. If k is too large, then it is clear that the overhead from the recomputation term will be too great. Conversely, if k is too small, then N_{prec} will grow as n gets further from $\tilde{t}(n)$. To strike a balance between these extremes, we dynamically keep track of the cumulative number of iterations above the baseline, N_{base}. As soon as this number exceeds the cost to compute a new preconditioner, N_{ms}, there should be a predicted benefit to recomputing at this time. We illustrate this approach in Figure 1.

6. Numerical examples. The following numerical examples seek to demonstrate the relative efficiency of the multiscale preconditioner with and without recomputation versus the nonpreconditioned interface problem. We shall compare three methods:

- **Method P1**—No interface preconditioner.
- **Method P2**—Multiscale preconditioner, computed once at initial time.

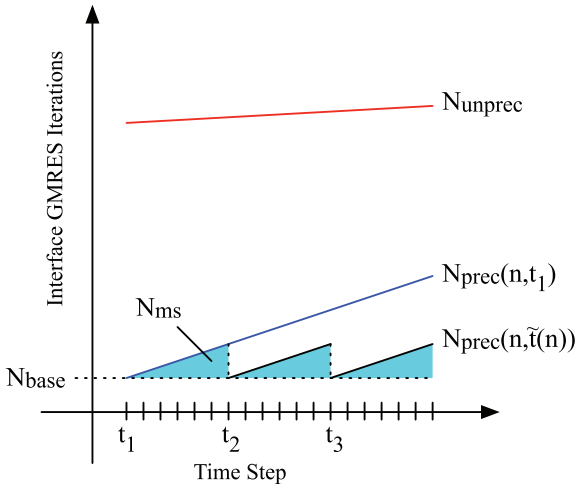

FIG. 1. *Heuristic for multiscale preconditioner recomputation.*

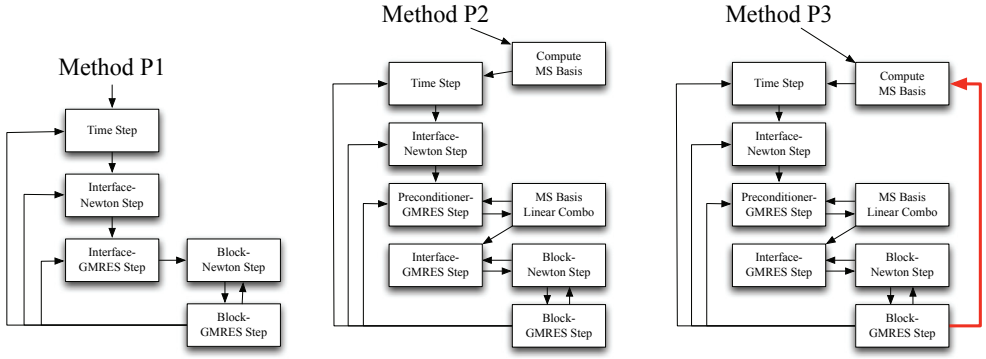

FIG. 2. *Flow charts for Methods* P1*,* P2*, and* P3*.*

- **Method P3**—Multiscale preconditioner, computed periodically using the heuristic described in section 5.1.

To illustrate the steps that each of these methods require, flow charts are given in Figure 2. Steps which may be parallelized appear in the right-most columns.

The reason for comparing our method to the unpreconditioned scheme (Method P1) is that, to the best of our knowledge, no efficient interface preconditioner has been developed for nonlinear problems such as single-phase slightly compressible flow or two-phase flow.

Since the multiscale mortar method allows a varying number of mortar degrees of freedom, the numerical results reported in this work utilize both matching and nonmatching interface grids. The former results in a discrete solution that is numerically equivalent to the single-domain formulation, given sufficient iterations of the domain decomposition. The latter results in an easier algebraic interface problem to solve if coarse mortars are chosen. Methods P1–P3 are equivalent algorithms for computing the given interface problems up to specified linear and nonlinear tolerances. In all cases, plots for pressure and saturation fields were virtually indistinguishable. In porous media applications, the quantity of interest is typically measured in well

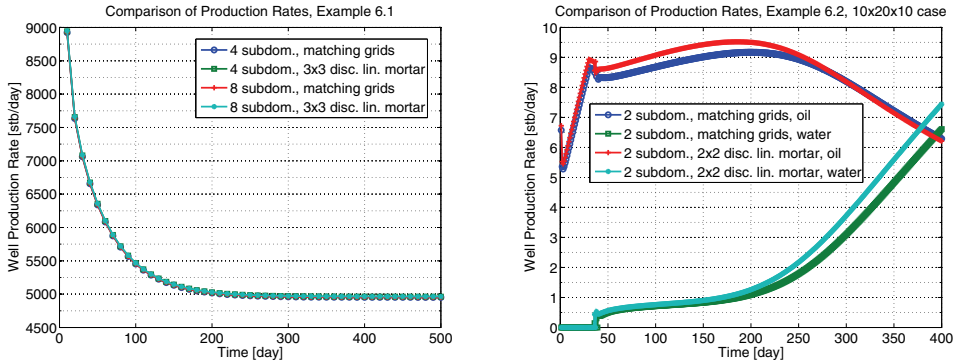

FIG. 3. *Comparison of injection well rates for the examples in sections* 6.1 *and* 6.2 *with matching and nonmatching grids.*

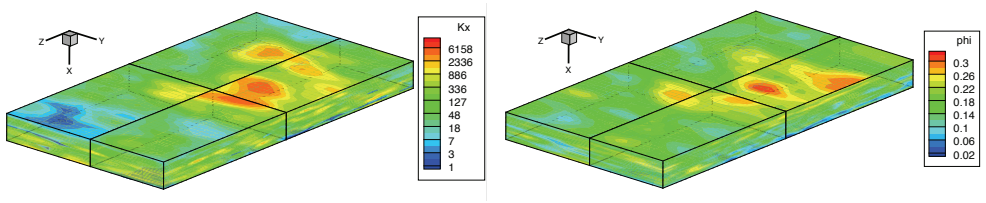

FIG. 4. *Upscaled SPE*10 *permeability* K_x *component (left) and porosity* ϕ *(right) for Example* 6.1.

production rates. Figure 3 shows that these rates are also nearly identical for matching and nonmatching grids. In the two-phase example, the water breakthrough time was at 36 days for all simulations. This evidence supports the assertion that solution quality is not drastically impacted when we weakly impose the interface conditions with relatively few mortar degrees of freedom.

Experiments were performed using the research code IPARS (implicit parallel accurate reservoir simulator) [23, 31], which was modified to implement the multiscale preconditioner. Our coordinate scheme follows the convention that x represents depth, and y and z represent arial directions. Runtimes are recorded by compiling the code without optimization, using Intel's *ifort* compiler and MKL library, and run in parallel using OpenMPI on a parallel cluster with hex core 2.93 GHz Intel Xeon X5670 processors.

6.1. Single phase: Heterogeneous case. In the first example we model slightly compressible single-phase flow using a heterogeneous permeability and porosity field that was taken from the SPE10 benchmark problem [13] on a domain Ω of size $168 \times 1200 \times 2000$ (ft) at a depth of 12000 (ft). It was upscaled to a $21 \times 15 \times 55$ grid, and decomposed into either $P = 4$ or $P = 8$ subdomains. The two choices are made to investigate the dependence of the behavior of the preconditioner on the number of processors. In general, the choice of number of subdomains (coarse scale) for practical problems is oftentimes driven by the physics of the problem. The domain decomposition may also be motivated by the need for different resolution in different parts of the domain through the use of nonmatching grids. Figure 4 shows the permeability and porosity on four subdomains. Other constants include fluid viscosity $\mu = 2$ (cp) and compressibility $c_w = 4E-5$ (1/psi). The initial fluid pressure is taken

to be $p(0) = 5000$ (psi). We model one injection well at 6000 (psi) and one production well at 4000 (psi) bottom hole pressure in a quarter five spot configuration, and use no-flow boundary conditions. We take constant time steps with $\Delta t = 10$ (days) and run the simulation until a steady state is reached after $T = 500$ (days). The resulting pressure field on four subdomains is shown in Figure 5.

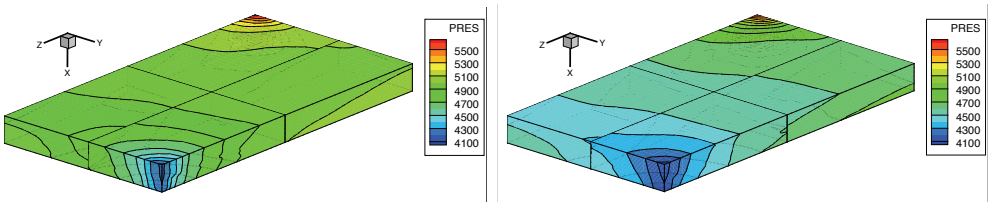

FIG. 5. *Pressure (psi) after* $t = 10$ *(left) and after* $t = 500$ *(right) days for Example* 6.1 *with matching grids. Plots for the multiscale solution were nearly identical.*

In our simulator, the interface and subdomain problems are solved with nested Newton-GMRES iterations. The innermost subdomain GMRES iterations are efficiently solved using a Line SOR preconditioner with 10 Gauss–Seidel smoothing steps (see [22]). Interface GMRES iterations will either be unpreconditioned or use the multiscale preconditioner. As seen in Figure 2, every application of the multiscale preconditioner requires an additional GMRES iteration wrapped around each Interface GMRES step. The tolerances which were chosen for these iterations are summarized in Table 1.

TABLE 1
Tolerances chosen for Example 6.1.

Finite difference Jacobian approximation	$1E-4$
Interface Newton tolerance	$1E-4$
Multiscale preconditioner GMRES tolerance	$1E-6$
Interface GMRES tolerance	$1E-6$
Subdomain Newton tolerance	$1E-6$
Subdomain GMRES tolerance	$1E-9$

We assign one processor to each subdomain, so that the multiscale preconditioner can be formed by solving a fixed number of subdomain problems in parallel. We run the test such that all interfaces have either discontinuous linear mortars with 3×3 elements or piecewise constant mortars on an interface grid that matches the traces of the subdomain grids. The latter results in a solution equivalent to the solution on fine scale matching grids with no mortars. We use both $P = 4$ and $P = 8$ subdomains, for a total of four tests. With $P = 4$ subdomains, the maximum number of interface degrees of freedom per subdomain is 72 for the mortar case and 756 for the matching grid case. With $P = 8$ subdomains, these numbers are 108 and 630. Note that with relatively few degrees of freedom, the interface problem is relatively easy to solve, and we compute a multiscale solution. With matching grids, the interface problem is much more difficult, and we compute a nonmultiscale solution. In the four tests, we run the simulation with and without the multiscale preconditioner and report the number of Interface GMRES iterations needed per time step in Figure 6.

In the multiscale simulation, the average number of Interface GMRES iterations was reduced from 79.7 to 3.8 for $P = 4$ subdomains, and from 151.7 to 3.9 for $P = 8$ subdomains. In the more challenging matching grid interface problem, the average

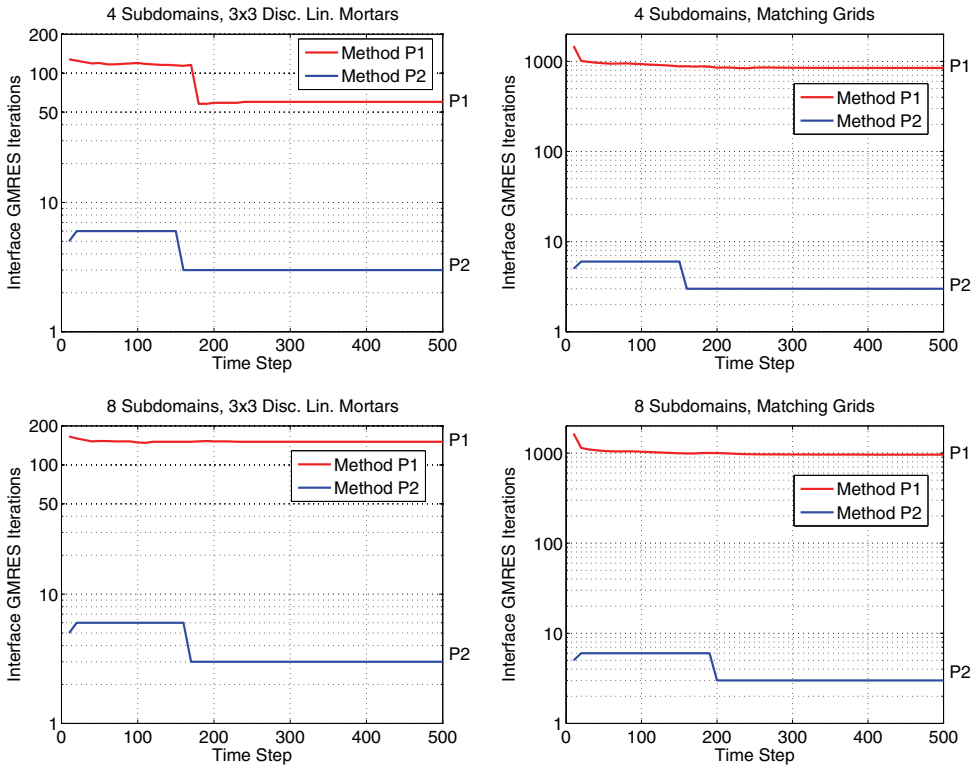

FIG. 6. *Interface GMRES iterations per time step for Example* 6.1.

TABLE 2
Runtimes for Example 6.1 *in seconds.*

		4 subdomains		8 subdomains	
		3×3 disc. lin. mortar	Matching grids	3×3 disc. lin. mortar	Matching grids
Method P1	Total time	1,226.1	11,672.6	1,876.2	8,344.3
Method P2	Total time	287.5	836.6	251.5	1,379.6
	Computing multiscale basis	16.8	161.4	8.9	42.4
	Applying preconditioner	122.7	553.0	145.9	1,231.7
	Linear combinations	10.6	146.2	12.7	158.0

number was reduced from 889.1 to 3.8 for $P = 4$ subdomains, and from 1004.9 to 4.1 for $P = 8$ subdomains. Note that the number of iterations for the preconditioned system did not increase with the number of subdomains, but the number of iterations for the unpreconditioned system did increase. As a result, the effectiveness of the preconditioner increased with the number of subdomains. In all four cases, no recomputation of the multiscale preconditioner was necessary, because its effectiveness remained constant throughout the simulation. This is most likely due to the relatively mild nonlinearity of the dynamics in the slightly compressible single-phase problem.

Table 2 shows that the reduction in Interface GMRES iterations has a significant impact on runtime. Each Interface GMRES iteration requires computing the solution to one nonlinear subdomain problem, so there are far fewer subdomain problems to be solved with Method P2 than with P1. However, Method P2 requires extra nonlinear

subdomain problems to be solved for each mortar degree of freedom when forming the multiscale basis. In time-dependent problems when computing the multiscale basis sparingly, this cost is small compared to the overall simulation time. Note that with roughly 10 (or 6) times as many interface degrees of freedom, the time to compute the multiscale basis is observed to increase by this amount in the matching grid case compared to the multiscale case.

Each time the multiscale preconditioner is applied, there are linear combinations to be performed, and the remainder of the time goes to interprocess communication. We can see that the linear combinations are a mere fraction of the time spent applying the preconditioner. For $P = 4$ subdomains, the average number of GMRES iterations required to apply the preconditioner was 60.7 in the multiscale case and 463.6 in the matching grid case. For $P = 8$ subdomains, this average was 77.1 in the multiscale case and 525.9 in the matching grid case. The linear combinations will also be 10 or 6 times as expensive due to the number of degrees of freedom. Together these two facts help to explain the increase in time for applying the preconditioner when comparing multiscale to matching grid.

Overall, this simulation shows that the multiscale preconditioner was highly effective for single-phase slightly compressible flow. For $P = 4$ subdomains, the total time was reduced by a factor of 4.26 in the multiscale case, and 13.95 in the matching grid case. For $P = 8$ subdomains, the total time was reduced by a factor of 7.45 in the multiscale case, and 6.04 in the matching grid case.

6.2. Two-phase: Flow around a barrier. In the second example we simulate flow around a low permeability barrier with the two-phase fully implicit model. The domain Ω is size $20 \times 200 \times 100$ (ft), and we run a total of five tests with Methods P1–P3. We split the domain into either $P = 2$ or $P = 4$ subdomains in the y direction. In addition, we compare two levels of fine scale resolution: $10 \times 20 \times 10$ elements and $10 \times 40 \times 20$ elements, as shown in Figure 7. The absolute permeability is a diagonal tensor, starting with $K_x = 50$ (md) for $\{y < 100\}$ and $K_x = 70$ (md) for $\{y \geq 100\}$. It is then modified by adding a staggered highly permeable channel with $K_x = 1000$ (md) and two low permeability barriers with $K_x = 1$ (md). The remaining components K_y and K_z are obtained by multiplying the K_x component by a factor of 2.

Other data includes fluid viscocities $\mu_w = 0.5$ and $\mu_o = 2.0$ (cp), fluid compressibilities $c_w = 3.3E-4$ and $c_o = 4.0E-3$ (1/psi), and the two relative permeability and one capillary pressure curves shown in Figure 8. The initial conditions are taken to be oil pressure $p_o(0) = 500$ (psi) and water saturation $S_w = 0.22$.

From an aerial perspective, water is injected at a well in the upper-left corner, and fluid is produced at a well in the lower-right corner, for a quarter five spot test with no-flow boundary conditions. Both wells are bottom hole pressure specified, with water injection starting at 505 (psi) and oil production starting at 495 (psi). These rates change linearly to 510 (psi) and 490 (psi), respectively, after $t = 30$ (days), after which both well rates remain constant. We take constant time steps with $\Delta t = 1$ (days) and run the simulation until a final time of $T = 400$ (days). The resulting oil concentration and water saturation fields are shown in Figure 9.

The setup for this simulation is much like the previous example; the same model tolerances were chosen as are listed in Table 1, and the type of parallelism employed is one processor per subdomain. On the $10 \times 20 \times 10$ grid, we use $P = 2$ subdomains, and the interface has either 2×2 discontinuous linear mortar or matching grids. The number of interface degrees of freedom per subdomain is 16 in the former and 100

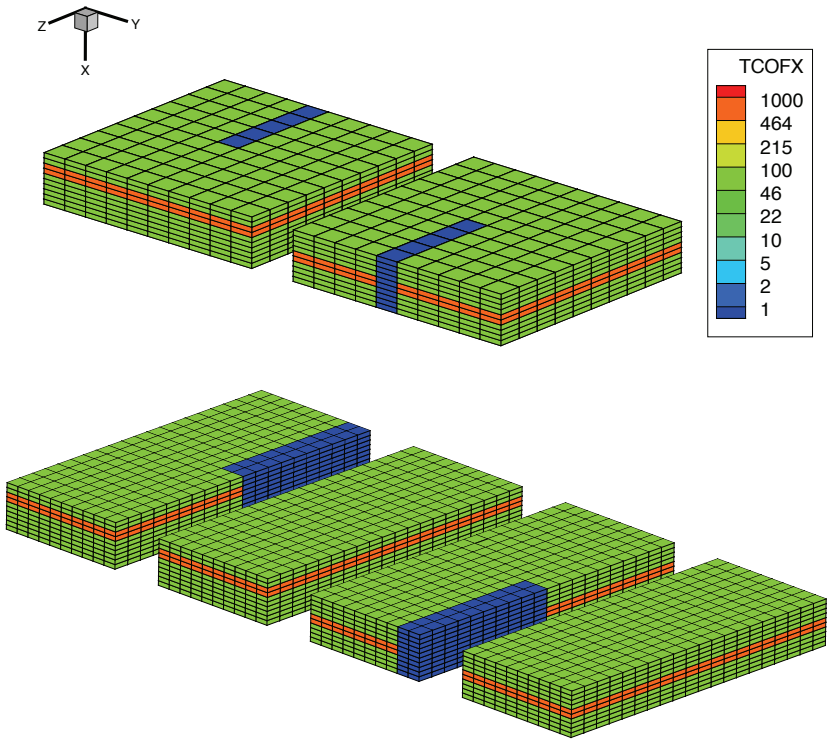

FIG. 7. *Permeability field K_x for Example* 6.2. *Top:* $10\times20\times10$ *with two subdomains. Bottom:* $10\times40\times20$ *with four subdomains.*

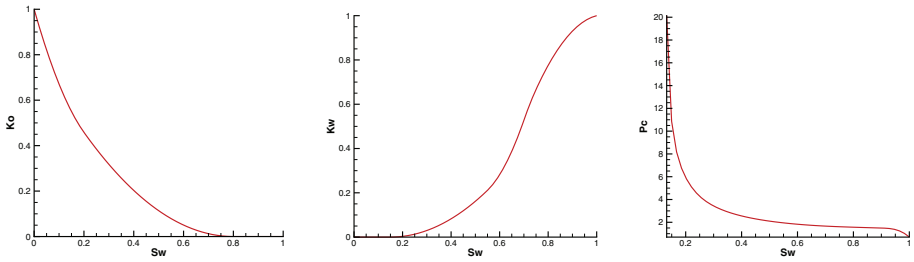

FIG. 8. *Oil relative permeability (left), water relative permeability (center), and capillary pressure (right) used for Example* 6.2.

in the latter. We compare computational results with and without the multiscale preconditioner, and note that in the two-phase simulation, the effectiveness of the multiscale preconditioner is observed to deteriorate as the interface Jacobians evolve in time. Hence, in this example we also employ Method P3 as described in section 5.1 and compare all three approaches. The number of Interface GMRES iterations per time step is reported in Figure 10.

In the multiscale simulation, the average number of interface GMRES iterations is 86.1 for the unpreconditioned problem. With Method P2, computing a single multiscale basis at the initial time step and reusing it as the multiscale preconditioner

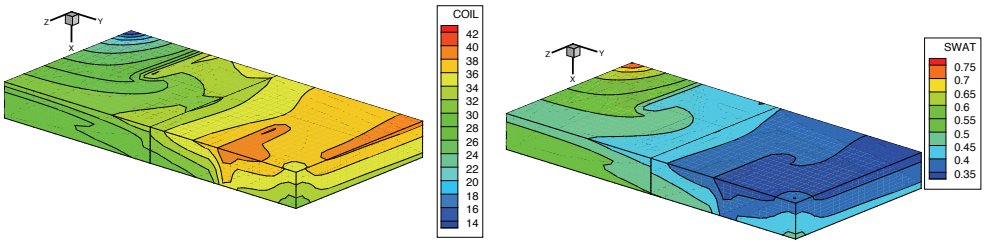

FIG. 9. *Oil concentration in lb/cu-ft (left) and water saturation (right) after* $T = 400$ *days for Example* 6.2.

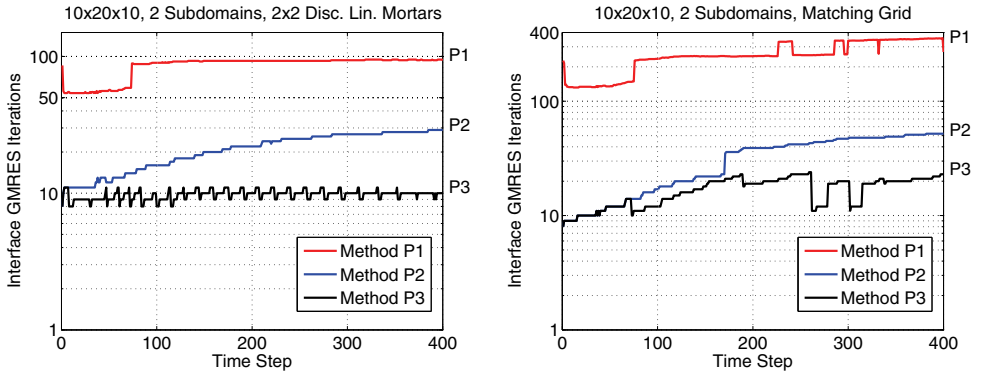

FIG. 10. *Interface GMRES iterations per time step for Example* 6.2 *on the* $10 \times 20 \times 10$ *grid.*

throughout the simulation brings the average number of interface GMRES iterations down to 21.2. With Method P3, a new multiscale basis is computed a total of 25 times throughout the simulation. These times are determined by when the total number of cumulative interface GMRES iterations above baseline preconditioning exceeds $N_{ms} = 16$. This brings the average number of interface GMRES iterations down to 9.7.

As expected, the interface problem in the matching grid simulation is more difficult to converge, so the effectiveness of the multiscale preconditioner is more pronounced. The average number of Interface GMRES iterations is 257.1 for the unpreconditioned problem. Method P2 brings this number down to 32.4. Method P3 further reduces it to 16.4. Note that in this case the multiscale basis is more expensive to recompute with $N_{ms} = 100$, and with this criteria it was only recomputed 10 times throughout the simulation.

Table 3 shows that the reduction in Interface GMRES iterations with the multiscale preconditioner does indeed have an effect on runtime in the two-phase simulation, but it is not nearly as effective as in the single-phase case. There are two reasons for the reduced effectiveness. First, the approximation of the interface Jacobian with a linear combination of multiscale basis functions is less effective in the more difficult two-phase model, so the multiscale preconditioner must be applied a greater number of times. Second, the cost in applying the multiscale preconditioner may be higher in highly nonlinear problems. Recall that the action of the preconditioner requires an interface GMRES iteration of its own; see Algorithm 3 and the discussion following it. While performing the linear combinations is relatively cheap, the interprocess

TABLE 3
Runtimes for Example 6.2 *in seconds on a coarse level.*

		2 subdomains	
		2×2 disc. lin. mortar	Matching grids
Method P1	Total time	7,262.8	19,144.2
Method P2	Total time	5,823.7	9,246.6
	Computing multiscale basis	4.3	26.8
	Applying preconditioner	3,793.8	6,001.1
	Linear combinations	153.4	289.5
Method P3	Total time	3,120.0	5,667.3
	Computing multiscale basis	124.1	194.9
	Applying preconditioner	1,912.2	3,482.2
	Linear combinations	73.7	173.2

communication may be expensive, as seen in the table.

In the multiscale simulation, the preconditioner took an average of 29 GMRES iterations to apply for every interface GMRES iteration. Method P2 reduced the runtime over Method P1 by a factor of 1.24. Note that in this case the time to compute a multiscale basis is negligible. With the additional effort of Method P3, the runtime was reduced over Method P1 by a factor of 2.32.

With a more challenging interface problem in the matching grid case, the results for the multiscale preconditioner are much better. In this case, the unpreconditioned problem is significantly more difficult, so the relative effectiveness of the multiscale preconditioner is more prominent—this despite the fact that it takes an average of 76 GMRES iterations to apply the multiscale preconditioner in this case. Using Method P2 reduced the total time by a factor of 2.07, and using Method P3 reduced the total time by a factor of 3.37 over Method P1.

Figure 11 shows the number of Interface GMRES iterations per time step for Example 6.2 on the $10 \times 40 \times 20$ grid. When comparing to the $10 \times 20 \times 10$ grid tests, the number of unpreconditoned iterations for Method P1 increases significantly for the matching grid case. However, the number of iterations for Methods P2 and P3 essentially remains unchanged. This indicates increased efficiency of the frozen Jacobian multiscale preconditioner when using finer grids.

Table 4 shows the corresponding effect on runtime for the $10 \times 40 \times 20$ grid simulations. When comparing to the $10 \times 20 \times 10$ grid test, runtimes for Method P1 are significantly higher. In contrast, the runtimes for Methods P2 and P3 increase more slowly, showing that our method is robust. From left to right, the reduction in runtime for Method P2 versus P1 is 2.81, 10.5, and 3.28. Method P3 has a greater reduction in runtime with 4.72, 14.1, and 6.49. Moreover, we see that increasing the number of subdomains for the $10 \times 40 \times 20$ grid test has a beneficial effect on runtime for Methods P2 and P3, but has a negative impact on runtime for Method P1.

7. Conclusions. We have demonstrated an effective approach for preconditioning the nonlinear interface problems associated with a mortar mixed finite element method. The frozen Jacobian multiscale preconditioner extends the concept of a multiscale flux basis, by recomputing it sparingly and reusing it throughout the simulation. The initial computational results for slightly compressible single-phase and two-phase flow are promising. This new type of physics-based preconditioner is notable because not many other techniques are known to precondition these types of systems. Although the multiscale preconditioner was extremely effective at reducing the number of Interface GMRES iterations, the communication overhead in applying

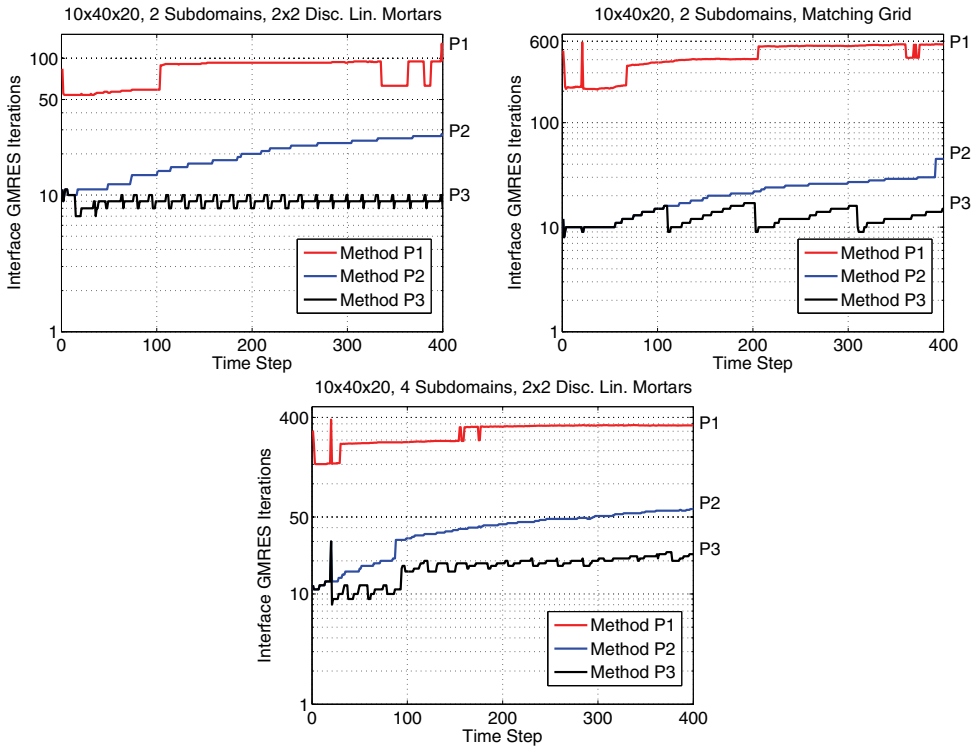

FIG. 11. *Interface GMRES iterations per time step for Example* 6.2 *on the* $10 \times 40 \times 20$ *grid.*

TABLE 4
Runtimes for Example 6.2 *in seconds on the* $10 \times 40 \times 20$ *grid.*

		2 subdomains		4 subdomains
		2×2 disc. lin. mortar	Matching grids	2×2 disc. lin. mortar
Method P1	Total time	28,755.3	128,138.9	32,341.2
Method P2	Total time	10,225.0	12,197.0	9,840.2
	Computing multiscale basis	23.2	264.3	6.7
	Applying preconditioner	2,278.8	3,007.0	4,929.6
	Linear combinations	34.6	156.0	284.2
Method P3	Total time	6,090.8	9,031.6	4,981.8
	Computing multiscale basis	623.4	1,101.2	131.1
	Applying preconditioner	1,177.8	1,911.7	2,398.9
	Linear combinations	20.3	105.5	149.3

the preconditioner is observed to be very costly. An important extension of this work could seek to mitigate this cost by combining the multiscale preconditioner with other preconditioning techniques.

We have focused on the MMMFEM to concisely describe the frozen Jacobian preconditioner. However, this approach could be extended to any multiscale method with localized subscale problems where a matrix could be formed that contains the coarse scale projection of the fine scale responses to each coarse scale degree of freedom based on the current state of the system. An exact or approximate inverse of this coarse scale matrix could be stored, or a secondary problem could be solved using this matrix as a preconditioner. This preconditioner could then be used across several nonlinear iterations and time steps.

REFERENCES

[1] J.E. Aarnes, *On the use of a mixed multiscale finite element method for greater flexibility and increased speed or improved accuracy in reservoir simulation*, Multiscale Model. Simul., 2 (2004), pp. 421–439.

[2] J.E. Aarnes, Y. Efendiev, and L. Jiang, *Mixed multiscale finite element methods using limited global information*, Multiscale Model. Simul., 7 (2008), pp. 655–676.

[3] J.E. Aarnes, S. Krogstad, and K.-A. Lie, *A hierarchical multiscale method for two-phase flow based upon mixed finite elements and nonuniform coarse grids*, Multiscale Model. Simul., 5 (2006), pp. 337–363.

[4] T. Arbogast, *Analysis of a two-scale, locally conservative subgrid upscaling for elliptic problems*, SIAM J. Numer. Anal., 42 (2004), pp. 576–598.

[5] T. Arbogast and K.J. Boyd, *Subgrid upscaling and mixed multiscale finite elements*, SIAM J. Numer. Anal., 44 (2006), pp. 1150–1171.

[6] T. Arbogast, L.C. Cowsar, M.F. Wheeler, and I. Yotov, *Mixed finite element methods on nonmatching multiblock grids*, SIAM J. Numer. Anal., 37 (2000), pp. 1295–1315.

[7] T. Arbogast, G. Pencheva, M.F. Wheeler, and I. Yotov, *A multiscale mortar mixed finite element method*, Multiscale Model. Simul., 6 (2007), pp. 319–346.

[8] T. Arobogast, C.N. Dawson, P.T. Keenan, M.F. Wheeler, and I. Yotov, *Enhanced cell-centered finite differences for elliptic equations on general geometry*, SIAM J. Sci. Comput., 19 (1998), pp. 404–425.

[9] F. Brezzi, J. Douglas, Jr., R. Durán, and M. Fortin, *Mixed finite elements for second order elliptic problems in three variables*, Numer. Math., 51 (1987), pp. 237–250.

[10] F. Brezzi, J. Douglas, Jr., M. Fortin, and L.D. Marini, *Efficient rectangular mixed finite elements in two and three space variables*, RAIRO Modél. Math. Anal. Numér., 21 (1987), pp. 581–604.

[11] F. Brezzi, J. Douglas, Jr., and L.D. Marini, *Two families of mixed finite elements for second order elliptic problems*, Numer. Math., 47 (1985), pp. 217–235.

[12] Z. Chen and T.Y. Hou, *A mixed multiscale finite element method for elliptic problems with oscillating coefficients*, Math. Comp., 72 (2003), pp. 541–576.

[13] M.A. Christie and M.J. Blunt, *Tenth SPE comparative solution project: A comparison of upscaling techniques*, SPE Reservoir Eval. Engrg., 4 (2001), pp. 308–317.

[14] Y.R. Efendiev, T.Y. Hou, and X.-H. Wu, *Convergence of a nonconforming multiscale finite element method*, SIAM J. Numer. Anal., 37 (2000), pp. 888–910.

[15] B. Ganis and I. Yotov, *Implementation of a mortar mixed finite element method using a multiscale flux basis*, Comput. Methods Appl. Mech. Engrg., 198 (2009), pp. 3989–3998.

[16] B. Ganis, I. Yotov, and M. Zhong, *A stochastic mortar mixed finite element method for flow in porous media with multiple rock types*, SIAM J. Sci. Comput., 33 (2011), pp. 1439–1474.

[17] T.Y. Hou and X.H. Wu, *A multiscale finite element method for elliptic problems in composite materials and porous media*, J. Comput. Phys., 134 (1997), pp. 169–189.

[18] T.Y. Hou, X.H. Wu, and Z. Cai, *Convergence of a multiscale finite element method for elliptic problems with rapidly oscillating coefficients*, Math. Comp., 68 (1999), pp. 913–943.

[19] T.J.R. Hughes, *Multiscale phenomena: Green's functions, the Dirichlet-to-Neumann formulation, subgrid scale models, bubbles and the origins of stabilized methods*, Comput. Methods Appl. Mech. Engrg., 127 (1995), pp. 387–401.

[20] T.J.R. Hughes, G.R. Feijóo, L. Mazzei, and J.B. Quincy, *The variational multiscale method—A paradigm for computational mechanics*, Comput. Methods Appl. Mech. Engrg., 166 (1998), pp. 3–24.

[21] P. Jenny, S.H. Lee, and H.A. Tchelepi, *Multi-scale finite-volume method for elliptic problems in subsurface flow simulation*, J. Comput. Phys., 187 (2003), pp. 47–67.

[22] C.T. Kelley, *Iterative Methods for Linear and Nonlinear Equations*, Frontiers in Appl. Math. 16, SIAM, Philadelphia, 1995.

[23] S. Lacroix, Y.V. Vassilevski, and M.F. Wheeler, *Iterative Solvers of the Implicit Parallel Accurate Reservoir Simulator (IPARS),* I: *Single Processor Case*, Technical Report 00-28, TICAM, University of Texas at Austin, Austin, TX, 2000.

[24] P. Le Tallec, J. Mandel, and M. Vidrascu, *A Neumann–Neumann domain decomposition algorithm for solving plate and shell problems*, SIAM J. Numer. Anal., 35 (1998), pp. 836–867.

[25] J. Mandel, *Balancing domain decomposition*, Comm. Numer. Methods Engrg., 9 (1993), pp. 233–241.

[26] J.C. Nédélec, *Mixed finite elements in* $\mathbf{R}^3$, Numer. Math., 35 (1980), pp. 315–341.

[27] D.W. Peaceman, *Fundamentals of Numerical Reservoir Simulation*, Elsevier, New York, 1977.

[28] G. Pencheva and I. Yotov, *Balancing domain decomposition for mortar mixed finite element methods*, Numer. Linear Algebra Appl., 10 (2003), pp. 159–180.
[29] M. Peszyńska, M.F. Wheeler, and I. Yotov, *Mortar upscaling for multiphase flow in porous media*, Comput. Geosci., 6 (2002), pp. 73–100.
[30] P.A. Raviart and J.M. Thomas, *A mixed finite element method for 2nd order elliptic problems*, in Mathematical Aspects of Finite Element Methods, Lecture Notes in Math. 606, Springer, Berlin, 1977, pp. 292–315.
[31] Y.V. Vassilevski, *Iterative Solvers for the Implicit Parallel Accurate Reservoir Simulator (IPARS),* II: *Parallelization Issues*, Technical Report 00-33, TICAM, University of Texas at Austin, Austin, TX, 2000.
[32] M.F. Wheeler, T. Wildey, and I. Yotov, *A multiscale preconditioner for stochastic mortar mixed finite elements*, Comput. Methods Appl. Mech. Engrg., 200 (2011), pp. 1251–1262.
[33] I. Yotov, *Mixed Finite Element Methods for Flow in Porous Media*, Ph.D. thesis, Computational and Applied Mathematics, Rice University, Houston, TX, 1996.
[34] I. Yotov, *A multilevel Newton–Krylov interface solver for multiphysics couplings of flow in porous media*, Numer. Linear Algebra Appl., 8 (2001), pp. 551–570.