SIAM J. SCI. COMPUT.
Vol. 36, No. 2, pp. A522–A542

A GLOBAL JACOBIAN METHOD FOR MORTAR DISCRETIZATIONS OF NONLINEAR POROUS MEDIA FLOWS*

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

Abstract. We describe a nonoverlapping domain decomposition algorithm for nonlinear porous media flows discretized with the multiscale mortar mixed finite element method. There are two main ideas: (1) linearize the global system in both subdomain and interface variables simultaneously to yield a single Newton iteration; and (2) algebraically eliminate subdomain velocities (and optionally, subdomain pressures) to solve linear systems for the 1st (or the 2nd) Schur complements. Solving the 1st Schur complement system gives the multiscale solution without the need to solve an interface iteration. Solving the 2nd Schur complement system gives a linear interface problem for a nonlinear model. The methods are less complex than a previously developed nonlinear mortar algorithm, which requires two nested Newton iterations and a forward difference approximation. Furthermore, efficient linear preconditioners can be applied to speed up the iteration. The methods are implemented in parallel, and a numerical study is performed to compare convergence behavior and parallel efficiency.

Key words. global Jacobian, nonlinear porous media flow, nonoverlapping domain decomposition, multiscale mortar mixed finite element, interface problem

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

DOI. 10.1137/130931837

1. Introduction.

There has been substantial success in coupling multiple scales, multiple physics, and multiple numerics using multiscale mortar methods [39, 38, 40, 42, 43, 30, 28, 7, 10, 19, 35]. This is a domain decomposition approach where the subdomains are discretized on possibly nonmatching grids by suitable numerical methods on a fine scale, and interface conditions are imposed on a coarse scale using mortar finite elements. In this work we focus on the multiscale mortar mixed finite element method (MMMFEM), where mixed finite element or related methods are employed for subdomain discretizations [7]. This framework allows for multiscale resolution whereby subdomain meshes may be adaptively refined around wells and channels, and subdomain interfaces may conform to complex geologic features such as faults and fractures. Moreover several physical models can be coupled together in separate physical regions, and computations can be performed locally with optimized numerical schemes.

*Submitted to the journal's Methods and Algorithms for Scientific Computing section August 5, 2013; accepted for publication (in revised form) January 16, 2014; published electronically March 25, 2014. 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, and also supported by NSF CDI grant DMS 0835745.

http://www.siam.org/journals/sisc/36-2/93183.html

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

‡Department of Mathematics and Systems Analysis, Aalto University, Aalto F1-00076, Finland (mojuntun@gmail.com).

§The Institute for Computational Engineering and Sciences (ICES), The University of Texas at Austin, TX 78712 (gergina@ices.utexas.edu, mfw@ices.utexas.edu).

¶Department of Mathematics, University of Pittsburgh, Pittsburgh, PA 15260 (yotov@math.pitt.edu). This author's work was partially supported by NSF grant DMS 1115856 and DOE grant DE-FG02-04ER25618.

Multiscale mortar methods share many similarities to other multiscale methods such as the variational multiscale method [24, 25, 4, 1, 5, 3], the multiscale finite element method [23, 22, 15, 13, 26, 2], and others that can be found in [34].

In many nonoverlapping domain decomposition methods, the tool of choice has been to reduce the resulting global multiscale system to a coarse scale interface problem solved by an iterative method. This is also known as a Steklov–Poincaré operator or the iterative substructuring approach [34, 31]. For linear problems, efficient interface solvers and preconditioners have been developed for mortar interface problems such as multigrid [40] and balancing [29]. However, in applying these ideas to nonlinear models, one must choose at which point(s) to linearize the system of coupled nonlinear equations. In this work we demonstrate that this choice strongly affects the complexity of the resulting algorithm and the speed of its convergence.

We refer to the previously developed algorithm to solve the MMMFEM with nonlinear model problems as the forward difference (FD) method [42, 43, 30]. In FD the interface problem is nonlinear and the first linearization is performed on the interface level, where the interface Jacobian is approximated using an FD. A second linearization occurs on the subdomain level with fixed interface data. This produces a complex algorithm with four nested iterations. Furthermore, one must carefully choose five progressively tighter tolerances, or else convergence may be lost. Standard acceleration techniques such as a forcing function, extrapolation, or preconditioning [16] may be applied but the overall complexity of the FD algorithm still remains.

This article offers a fresh perspective to address the issues that have inhibited the full potential of the MMMFEM with nonlinear models. We refer to our methodology as the global Jacobian (GJ) method, and in this work we describe its application to a nonlinear single-phase slightly compressible porous media flow model. There are two main ideas to our approach.

1. We linearize the global system in both subdomain and interface variables simultaneously to yield a single Newton iteration.
2. We algebraically eliminate subdomain velocities (and optionally, subdomain pressures) to solve linear systems for the 1st (or the 2nd) Schur complements.

Using these ideas, we form two algorithms that have competitive parallel performance for nonlinear model problems. In the 1st Schur complement formulation, called the GJ method, we form a matrix corresponding to a global multiscale linear system. In this case, it is easy to apply a generic solver library. This algorithm doesn't involve the solution of an interface problem, which is similar to the enhanced velocity method for handling nonmatching grids [37]. In the 2nd Schur complement formulation, called the GJS (global Jacobian Schur) method, we have a matrix-free, linear interface problem to solve. In this case, it is possible to apply preconditioners developed for linear interface problems, such as a balancing preconditioner [29] or reduce the cost of each interface iteration by the construction of a multiscale flux basis [17]. Although the second algorithm involves an extra iteration, it reduces the interprocessor communication and exhibits superior parallel scalability. Both algorithms have reduced complexity compared to the FD method, which has two nested nonlinear iterations. Figure 1 illustrates the differences in these algorithms.

The outline of this article is as follows: in section 2, the GJ method is formulated for the 1st and 2nd Schur complement systems with a single-phase, slightly compressible model problem; in section 3, the GJ method is compared to the FD method; in section 4, numerical results are presented for both linear and nonlinear example problems to demonstrate convergence behavior and parallel scalability. The two proposed

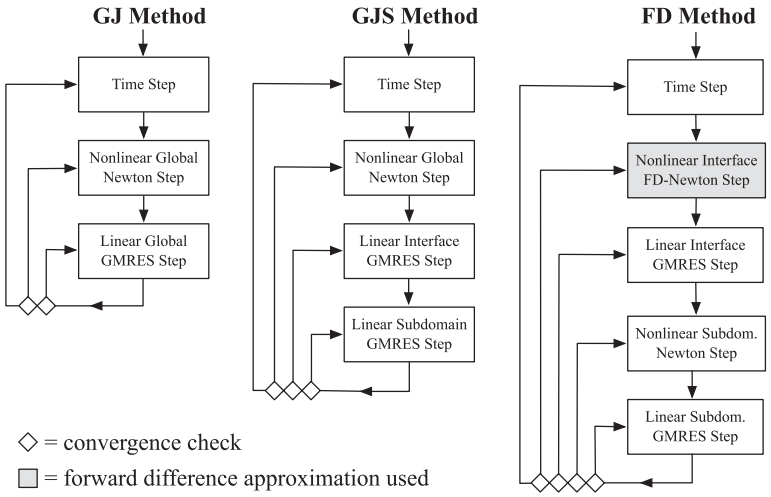

FIG. 1. *Flow charts comparing the GJ, GJS, and FD methods.*

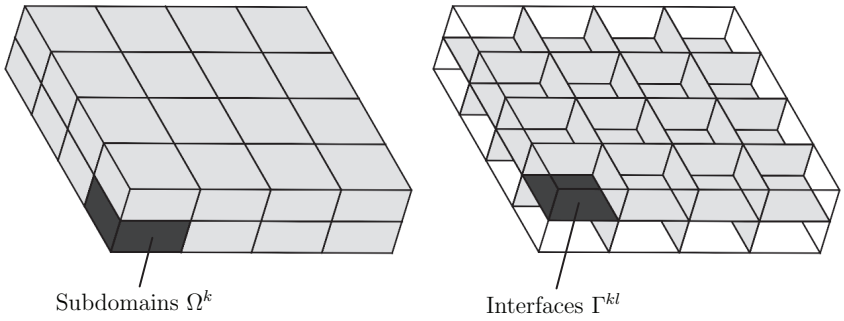

FIG. 2. *Illustration of nonoverlapping domain decomposition.*

methods are compared to the FD method as well as single domain multiple processor data decomposition.

2. Formulation of the global Jacobian (GJ) method. In this section, we describe the formulation of the GJ method for single-phase, slightly compressible flow in porous media with the MMMFEM. To formulate our method, five steps are performed in the following order:

1. discretize in time using the backward Euler method;
2. discretize in space using the MMMFEM;
3. linearize the global problem simultaneously with respect to pressure, velocity, and mortar variables;
4. eliminate subdomain velocities to form the 1st Schur complement system;
5. (optional) eliminate subdomain pressures to form the 2nd Schur complement system.

2.1. Single-phase slightly compressible flow model. Consider a time interval $[0, T]$, along with a spatial domain, $\Omega \subset \mathbb{R}^d$, $d = 2$ or 3 with boundary $\partial\Omega$ and outward normal $\boldsymbol{n}$. As Figure 2 shows, the spatial domain is decomposed into N_Ω

nonoverlapping subdomains such that

$$\overline{\Omega} = \bigcup_{k=1}^{N_\Omega} \overline{\Omega^k}, \qquad \Omega^k \cap \Omega^l = \emptyset \text{ when } k \neq l.$$

Subdomain interfaces are denoted by $\Gamma^{kl} = \Gamma^{lk} = \partial\Omega^k \cap \partial\Omega^l$ and $\Gamma = \bigcup_{1\leq k<l\leq N_\Omega} \Gamma^{kl}$.

In this setting, the model for single-phase slightly compressible flow through a porous medium is given by, for $k = 1, \ldots, N_\Omega$,

$$\boldsymbol{u} = -\frac{K}{\mu}\rho\,(\nabla p - \rho \boldsymbol{g}) \quad \text{in } \Omega^k \times (0,T], \tag{2.1a}$$

$$\frac{\partial}{\partial t}(\phi\rho) + \nabla\cdot\boldsymbol{u} = q \quad \text{in } \Omega^k \times (0,T], \tag{2.1b}$$

$$\boldsymbol{u}\cdot\boldsymbol{n} = 0 \quad \text{on } \partial\Omega \times (0,T], \tag{2.1c}$$

$$p = p_0 \quad \text{on } \Omega \times \{t = 0\}, \tag{2.1d}$$

$$p = \lambda \quad \text{on } \Gamma \times (0,T], \tag{2.1e}$$

$$\boldsymbol{u}^k\cdot\boldsymbol{n}^k + \boldsymbol{u}^l\cdot\boldsymbol{n}^l = 0 \quad \text{on } \Gamma^{kl} \times (0,T]. \tag{2.1f}$$

The subsurface flow is characterized by Darcy's law (2.1a) together with a conservation of mass (2.1b) over every subdomain. The unknowns are fluid velocity $\boldsymbol{u}(\boldsymbol{x},t)$, fluid pressure $p(\boldsymbol{x},t)$, and Lagrange multiplier $\lambda(\boldsymbol{x},t)$, introduced in (2.1e), that has the physical meaning of interface pressure. Given data include the porosity $\phi(\boldsymbol{x})$, the mass flux source term $q(\boldsymbol{x},t)$, the second order permeability tensor $K(\boldsymbol{x})$, the fluid viscosity μ, and the gravitational acceleration vector $\boldsymbol{g}$. For simplicity in the presentation, a no-flow external boundary condition (2.1c) is assumed, with a remark that more general boundary conditions can be handled with minor modification. The initial condition (2.1d) is assumed to be under hydrostatic equilibrium. In (2.1f) a conservation of mass is enforced over the entire domain. The $\boldsymbol{n}^k$ denotes the outer normal of Ω^k and $\boldsymbol{u}^k = \boldsymbol{u}|_{\Omega^k}$.

The model is closed with an exponential equation of state that specifies fluid density ρ as a function of fluid pressure:

$$\rho(p) = \rho_{\text{ref}}\, e^{c(p-p_{\text{ref}})}. \tag{2.2}$$

Given data include the reference density ρ_{ref}, the reference pressure p_{ref}, and the fluid compressibility constant c. The permeability tensor is assumed to be uniformly bounded and positive definite, i.e., there exist constants $0 < k_0 \leq k_1 < \infty$ such that

$$k_0\|\xi\|^2 \leq \xi^T K(\boldsymbol{x})\xi \leq k_1\|\xi\|^2 \quad \forall \xi \in \mathbb{R}^d,\ \forall \boldsymbol{x} \in \Omega, \tag{2.3}$$

where $\|\cdot\|$ is the Euclidean vector norm.

2.2. Fully discrete formulation. On each subdomain Ω^k, choose an independent spatial discretization $\mathcal{T}_h^k$ with fine scale characteristic mesh spacing h. These meshes are allowed to be nonmatching on subdomain interfaces. To simplify the presentation, we choose to discretize the problem with the lowest order Raviart–Thomas elements (RT_0) [32] on orthogonal bricks (or rectangles) for $d = 3$ (or 2):

$$\boldsymbol{V}_h^k = \{\boldsymbol{v} \in H(\text{div};\Omega^k) \,:\, \boldsymbol{v}|_E \in RT_0(E)\ \forall E \in \mathcal{T}_h^k\,,\ \boldsymbol{v}\cdot\boldsymbol{n} = 0 \text{ on } \partial\Omega\},$$
$$W_h^k = \{w \in L^2(\Omega^k) \,:\, w|_E \in \mathcal{P}_0(E)\ \forall E \in \mathcal{T}_h^k\},$$

in which $\mathcal{P}_m(E)$ denotes the polynomials of order m on element E and the i-component of a vector in $RT_0(E)$ is linear in x_i and constant in x_j, $j \neq i$.

We note that the algorithms apply to any of the stable mixed finite element spaces of arbitrary order on simplicial, quadrilateral, or hexahedral elements.

On each interface Γ^{kl}, choose an independent spatial discretization $\mathcal{G}_H^{kl}$ on rectangles (or intervals) for $(d-1) = 2$ (or 1) with coarse scale characteristic mesh spacing H. Let M_H^{kl} be a mortar space on $\mathcal{G}_H^{kl}$ consisting of continuous or discontinuous polynomials:

$$M_H^{kl} = \{\mu \in L^2(\Gamma^{kl}) \,:\, \mu|_E \in \mathcal{P}_m(E)\ \forall E \in \mathcal{G}_H^{kl},\, m \geq 0\}.$$

The global velocity, pressure, and Lagrange multiplier spaces are

$$\boldsymbol{V}_h = \bigoplus_{k=1}^{N_\Omega} \boldsymbol{V}_h^k, \qquad W_h = \bigoplus_{k=1}^{N_\Omega} W_h^k, \qquad M_H = \bigoplus_{1 \leq k < l \leq N_\Omega} M_H^{kl}.$$

In the incompressible case $c = 0$ we further require that functions in W_h have mean value zero in Ω.

Choose a temporal discretization $0 = t^0 < t^1 < \cdots < t^{N_T} = T$, with $\delta t^n = t^n - t^{n-1}$. To simplify notation, the time index n is suppressed except for ρ^{n-1}, which denotes the known density at the previous time step.

Employing the backward Euler method for time integration, with the implicit treatment of both density and source functions, the fully discrete formulation of the MMMFEM in the case of the nonlinear model (2.1a)–(2.1f) is as follows. For time levels $n = 1, \ldots, N_T$, find $(\boldsymbol{u}^k, p^k, \lambda^{kl}) \in \boldsymbol{V}_h^k \times W_h^k \times M_H^{kl}$ such that for subdomains Ω^k, $k = 1, \ldots, N_\Omega$,

$$\left(\mu\ \rho^{-1} K^{-1} \boldsymbol{u}^k, \boldsymbol{v}^k\right)_k - (p^k, \nabla \cdot \boldsymbol{v}^k)_k - (\rho \boldsymbol{g}, \boldsymbol{v}^k)_k + \sum_{l=1,\, l \neq k}^{N_\Omega} \langle \lambda^{kl}, \boldsymbol{v}^k \cdot \boldsymbol{n}^k \rangle_{kl} = 0, \tag{2.4a}$$

$$-(\phi\rho, w^k)_k - \left(\delta t\, \nabla \cdot \boldsymbol{u}^k, w^k\right)_k + \left(\phi \rho^{n-1} + \delta t\, q, w^k\right)_k = 0, \tag{2.4b}$$

and for interfaces Γ^{kl}, $1 \leq k < l \leq N_\Omega$,

$$\langle \boldsymbol{u}^k \cdot \boldsymbol{n}^k + \boldsymbol{u}^l \cdot \boldsymbol{n}^l, \eta^{kl} \rangle_{kl} = 0 \tag{2.4c}$$

for all $(\boldsymbol{v}^k, w^k, \eta^{kl}) \in \boldsymbol{V}_h^k \times W_h^k \times M_H^{kl}$. The $(\cdot, \cdot)_k$ denotes the L^2 inner product over subdomain Ω^k and $\langle \cdot, \cdot \rangle_{kl}$ the L^2 inner product over interface Γ^{kl}.

Let $\{\boldsymbol{v}_i^k\}_{i=1}^{N_{\boldsymbol{u}}^k}$, $\{w_i^k\}_{i=1}^{N_p^k}$, and $\{\eta_i^{kl}\}_{i=1}^{N_\lambda^{kl}}$ be the finite element basis functions for $\boldsymbol{V}_h^k$, W_h^k, and M_H^{kl}, respectively. We can now express the unknowns as the linear combinations

$$\boldsymbol{u}^k = \sum_{i=1}^{N_{\boldsymbol{u}}^k} U_i^k \boldsymbol{v}_i^k, \quad p^k = \sum_{i=1}^{N_p^k} P_i^k w_i^k, \quad \text{and } \lambda^{kl} = \sum_{i=1}^{N_\lambda^{kl}} \Lambda_i^{kl} \eta_i^{kl},$$

where $U^k \in \mathbb{R}^{N_{\boldsymbol{u}}^k}$, $P^k \in \mathbb{R}^{N_p^k}$, and $\Lambda^{kl} \in \mathbb{R}^{N_\lambda^{kl}}$ are the Euclidean vectors of unknown coefficients.

Let $N_{\boldsymbol{u}} = \sum_{k=1}^{N_\Omega} N_{\boldsymbol{u}}^k$, $N_p = \sum_{k=1}^{N_\Omega} N_p^k$, and $N_\lambda = \sum_{1\le k<l\le N_\Omega} N_\lambda^{kl}$. By ordering the unknowns according to subdomains and interfaces, define the global coefficient vectors $U \in \mathbb{R}^{N_{\boldsymbol{u}}}$, $P \in \mathbb{R}^{N_p}$, and $\Lambda \in \mathbb{R}^{N_\lambda}$ as

$$U = \begin{pmatrix} U^1 \\ \vdots \\ U^{N_\Omega} \end{pmatrix}, \quad P = \begin{pmatrix} P^1 \\ \vdots \\ P^{N_\Omega} \end{pmatrix}, \quad \text{and} \quad \Lambda = \begin{pmatrix} \Lambda^{12} \\ \vdots \\ \Lambda^{(N_\Omega-1)N_\Omega} \end{pmatrix}. \tag{2.5}$$

In the above, without loss of generality, we have assumed that Γ_{12} and $\Gamma_{(N_\Omega-1)N_\Omega}$ are nonempty.

Under these representations, we recast the nonlinear equations (2.4a)–(2.4c) as residual equations. For Ω_k, $k = 1, \dots, N_\Omega$, and $j = 1, \dots, N_{\boldsymbol{u}}^k$, define

$$\begin{aligned} F_j^k(U,P,\Lambda) = &\left(\mu\rho_0^{-1} e^{-c\sum_{i=1}^{N_p^k} P_i^k w_i^k} K^{-1} \sum_{i=1}^{N_{\boldsymbol{u}}^k} U_i^k \boldsymbol{v}_i^k,\ \boldsymbol{v}_j^k\right)_k - \left(\sum_{i=1}^{N_p^k} P_i^k w_i^k,\ \nabla\cdot\boldsymbol{v}_j^k\right)_k \\ &- \left(\rho_0 e^{c\sum_{i=1}^{N_p^k} P_i^k w_i^k}\boldsymbol{g},\ \boldsymbol{v}_j^k\right)_k + \sum_{l=1,\, l\neq k}^{N_\Omega} \left\langle \sum_{i=1}^{N_\lambda^{kl}} \Lambda_i^{kl}\eta_i^{kl},\ \boldsymbol{v}_j^k\cdot\boldsymbol{n}^k\right\rangle_{kl}. \end{aligned} \tag{2.6a}$$

For Ω_k, $k = 1, \dots, N_\Omega$, and $j = 1, \dots, N_p^k$, define

$$\begin{aligned} G_j^k(U,P) = &-\left(\phi\rho_0 e^{c\sum_{i=1}^{N_p^k} P_i^k w_i^k},\ w_j^k\right)_k - \left(\delta t\,\nabla\cdot\sum_{i=1}^{N_{\boldsymbol{u}}^k} U_i^k\boldsymbol{v}_i^k,\ w_j^k\right)_k \\ &+ \left(\phi\rho^{n-1} + \delta t\, q,\ w_j^k\right)_k. \end{aligned} \tag{2.6b}$$

For Γ^{kl}, $1 \le k < l \le N_\Omega$, and $j = 1, \dots, N_\lambda^{kl}$, define

$$H_j^{kl}(U) = \left\langle \sum_{i=1}^{N_{\boldsymbol{u}}^k} U_i^k\boldsymbol{v}_i^k\cdot\boldsymbol{n}^k + \sum_{i=1}^{N_{\boldsymbol{u}}^l} U_i^l\boldsymbol{v}_i^l\cdot\boldsymbol{n}^l,\ \eta_j^{kl}\right\rangle_{kl}. \tag{2.6c}$$

By ordering F, G, and H according to (2.5), at time level n the residual equations are

$$F(U,P,\Lambda) = 0, \tag{2.7a}$$

$$G(U,P) = 0, \tag{2.7b}$$

$$H(U) = 0, \tag{2.7c}$$

which is a set of $N_{\boldsymbol{u}} + N_p + N_\lambda$ nonlinear equations for the unknown vector (U,P,Λ) of the same size.

2.3. Linearization of the global problem. Next we compute the partial derivatives of the residual equations with respect to each unknown, by application

of the chain rule:

$$A^k_{ji} := \frac{\partial F^k_j}{\partial U^k_i}(U,P,\Lambda) = (\mu\rho^{-1}K^{-1}\boldsymbol{v}^k_i, \boldsymbol{v}^k_j)_k,\ j,i = 1,\dots,N^k_{\boldsymbol{u}}, \tag{2.8a}$$

$$\tilde{B}^k_{ji} := \frac{\partial F^k_j}{\partial P^k_i}(U,P,\Lambda) = -(cw^k_i\mu\rho^{-1}K^{-1}\boldsymbol{u}^k, \boldsymbol{v}^k_j)_k - (w^k_i, \nabla\cdot\boldsymbol{v}^k_j)_k$$

$$- \left(cw^k_i\rho\boldsymbol{g}, \boldsymbol{v}^k_j\right)_k,\ j = 1,\dots,N^k_{\boldsymbol{u}},\ i = 1,\dots,N^k_p, \tag{2.8b}$$

$$L^{kl}_{ji} := \frac{\partial F^k_j}{\partial \Lambda^{kl}_i}(U,P,\Lambda) = \langle \eta^{kl}_i, \boldsymbol{v}^k_j\cdot\boldsymbol{n}^k\rangle_{kl},\ j = 1,\dots,N^k_{\boldsymbol{u}},\ i = 1,\dots,N^{kl}_\lambda, \tag{2.8c}$$

$$C^k_{ji} := \frac{\partial G^k_j}{\partial U^k_i}(U,P) = -(\delta t\,\nabla\cdot\boldsymbol{v}^k_i, w^k_j)_k,\ j = 1,\dots,N^k_p,\ i = 1,\dots,N^k_{\boldsymbol{u}}, \tag{2.8d}$$

$$D^k_{ji} := \frac{\partial G^k_j}{\partial P^k_i}(U,P) = -(cw^k_i\phi\rho, w^k_j)_k,\ j,i = 1,\dots,N^k_p. \tag{2.8e}$$

Note also that

$$L^{kl}_{ij} = \langle \boldsymbol{v}^k_i\cdot\boldsymbol{n}^k, \eta^{kl}_j\rangle_{kl} = \frac{\partial H^{kl}_j}{\partial U^k_i}(U),\ j = 1,\dots,N^{kl}_\lambda,\ i = 1,\dots,N^k_{\boldsymbol{u}}.$$

In our numerical methods for slightly compressible, single-phase flow, we drop the two terms with compressibility c in (2.8b) to define a nearly symmetric Jacobian matrix with improved spectrum, such that

$$B^k = \delta t^{-1}(C^k)^T.$$

This commonly used technique is an approximation to the Jacobian (matrix). This technique works well for the slightly compressible flow model because c is small. The residual vector (right-hand side) is still computed exactly. Consistent with the theory on inexact Newton's methods [27], when convergence is achieved, the solution to this nonlinear system will coincide with the full Newton method.

Define

$$A = \begin{pmatrix} A^1 & & \\ & \ddots & \\ & & A^{N_\Omega} \end{pmatrix},\ B = \begin{pmatrix} B^1 & & \\ & \ddots & \\ & & B^{N_\Omega} \end{pmatrix},$$

$$D = \begin{pmatrix} D^1 & & \\ & \ddots & \\ & & D^{N_\Omega} \end{pmatrix},\ \text{and } L = \begin{pmatrix} L^{12} \\ \vdots \\ L^{(N_\Omega-1)N_\Omega} \end{pmatrix}. \tag{2.9}$$

These blocks form the GJ matrix and achieve linearization of the problem (2.4a)–(2.4c). The linear system for the global inexact Newton step is given by

$$\begin{bmatrix} A & B & L \\ \delta t B^T & D & 0 \\ L^T & 0 & 0 \end{bmatrix} \begin{bmatrix} \delta U \\ \delta P \\ \delta\Lambda \end{bmatrix} = - \begin{bmatrix} F \\ G \\ H \end{bmatrix}. \tag{2.10}$$

2.4. Forming the 1st and 2nd Schur complements. There are a wide variety of numerical techniques for solving saddle point systems [11]. In this work, our

approach is to form a Schur complement of the saddle point system (2.10) by algebraically eliminating the velocity. A similar idea is used in the formulation of the hybrid mixed finite element method, where velocities are eliminated on the element level and the Lagrange multipliers are defined on element faces (or edges) [9]. In our case the velocity is eliminated on the subdomain level. Observe that in (2.9) the matrices A, B, and D are all block diagonal, with blocks grouping the unknowns on each subdomain. The matrix L is very sparse, since it merely couples subdomain variables adjacent to interfaces with Lagrange multipliers.

The modeling assumptions in this work are the use of structured orthogonal brick elements and a positive definite diagonal permeability tensor K. Therefore with the application of the trapezoidal-midpoint quadrature rule [8], the mass matrix A is diagonal and trivial to invert. The case when K is a full tensor can be handled using the expanded mixed finite element method [8], and the case when grids are general quadrilaterals or hexahedra can be handled using the multipoint flux mixed finite element method [41]. In these cases, the matrix A is block diagonal with blocks grouping the unknowns associated with mesh vertices, and LU factorization can be inexpensively applied to each block, and possibly recycled in time. Furthermore, this approach can be generalized to arbitrary mixed finite element discretizations, leading to fuller sparsity patterns of A, and velocity elimination can be handled with a matrix-free approach whenever the action of A^{-1} is required. These cases are natural extensions to, but beyond, the scope of this work.

Therefore, using the substitution $\delta U = -A^{-1}(B\ \delta P + L\ \delta\Lambda + F)$ and multiplying the equations by -1, the 1st Schur complement system for the Newton step is given by

$$\begin{aligned} &\begin{bmatrix} \delta t\ B^T A^{-1} B - D & \delta t\ B^T A^{-1} L \\ L^T A^{-1} B & L^T A^{-1} L \end{bmatrix} \begin{bmatrix} \delta P \\ \delta\Lambda \end{bmatrix} = \begin{bmatrix} G - \delta t\ B^T A^{-1} F \\ H - L^T A^{-1} F \end{bmatrix} \\ \iff \quad &J \begin{bmatrix} \delta P \\ \delta\Lambda \end{bmatrix} = \begin{bmatrix} J_{PP} & J_{P\Lambda} \\ J_{\Lambda P} & J_{\Lambda\Lambda} \end{bmatrix} \begin{bmatrix} \delta P \\ \delta\Lambda \end{bmatrix} = \begin{bmatrix} R_P \\ R_\Lambda \end{bmatrix}. \end{aligned} \tag{2.11}$$

We refer to solving the 1st Schur complement system (2.11) as the GJ method. Note that this is a nonoverlapping domain decomposition algorithm for mortar mixed finite element discretizations where mortar and subdomain variables are computed simultaneously without the need to solve a secondary interface iteration. We give a pseudocode for the implementation of the corresponding Newton-GMRES algorithm in Algorithm 1. Since the matrix J is positive definite (see Lemma 2.1 below), the generalized minimum residual method (GMRES) [33, 27] is used to solve the linear systems. The method is fully parallelizable by assembling and storing pieces of this linear system on separate processors. The GMRES solver and the application of a preconditioner are both implemented in parallel using distributed matrix-vector operations.

In (2.11), J_{PP} is a block diagonal matrix, and each decoupled block of subdomain pressure unknowns is invertible; see Lemma 2.2 below. This allows us to perform the (optional) fifth step. Using the substitution $\delta P = J_{PP}^{-1}(R_P + J_{\Lambda P}\delta\Lambda)$, the 2nd Schur complement system for the Newton step is given by

$$\begin{aligned} &(J_{\Lambda\Lambda} - J_{\Lambda P} J_{PP}^{-1} J_{P\Lambda})\ \delta\Lambda = R_\Lambda - J_{\Lambda P} J_{PP}^{-1} R_P \\ \iff \quad &J_S\ \delta\Lambda = R_S. \end{aligned} \tag{2.12}$$

Note that J_S is positive definite; see Lemma 2.3 below. We refer to solving the 2nd Schur complement system (2.12) as the GJS method, and give pseudocode for

Algorithm 1. The GJ method.

{Superscript n denotes time index and n, k Newton index k at time n.}
Given $[P^0, \Lambda^0]$, {Initial condition}
for $n = 1, \ldots, N_T$ {Time index} **do**
  $[P^{n,0}, \Lambda^{n,0}] = [P^{n-1}, \Lambda^{n-1}]$ {Initial guess for Newton}
  for $k = 0, \ldots,$ NEWT_MAX {Newton index} **do**
    $R^{n,k} = [R_P, R_\Lambda](P^{n,k}, \Lambda^{n,k})$ {Form Newton residual}
    if ($\|R^{n,k}\| <$ NEWT_TOL) **then break** k-loop {Check nonlinear convergence}
    $J^{n,k} = [J_{PP}, J_{P\Lambda}; J_{\Lambda P}, J_{\Lambda\Lambda}](P^{n,k}, \Lambda^{n,k})$ {Form 1st Schur complement}
    $[\delta P, \delta\Lambda] =$ **gmres**$(J^{n,k}, R^{n,k})$ {Solve global linear system}
    $[P^{n,k+1}, \Lambda^{n,k+1}] = [P^{n,k} + \delta P, \Lambda^{n,k} + \delta\Lambda]$ {Increment unknowns}
  end for
  $[P^n, \Lambda^n] = [P^{n,k}, \Lambda^{n,k}]$ {Newton has converged}
end for

Algorithm 2. The GJS method.

{Superscript n denotes time index and n, k Newton index k at time n.}
Given $[P^0, \Lambda^0]$, {Initial condition}
for $n = 1, \ldots, N_T$ {Time index} **do**
  $[P^{n,0}, \Lambda^{n,0}] = [P^{n-1}, \Lambda^{n-1}]$ {Initial guess for Newton}
  for $k = 0, \ldots,$ NEWT_MAX {Newton index} **do**
    $R^{n,k} = [R_P, R_\Lambda](P^{n,k}, \Lambda^{n,k})$ {Form Newton residual}
    if ($\|R^{n,k}\| <$ NEWT_TOL) **then break** k-loop {Check nonlinear convergence}
    $J^{n,k} = [J_{PP}, J_{P\Lambda}; J_{\Lambda P}, J_{\Lambda\Lambda}](P^{n,k}, \Lambda^{n,k})$ {Form 1st Schur complement}
    function $J_S(\xi)$ {**return** $J_{\Lambda\Lambda}\xi - J_{\Lambda P}$ **gmres**$(J_{PP}, J_{P\Lambda}\xi)$}
      {Define action of 2nd Schur complement by solving linear subdomain problems}
    $R_S = R_\Lambda - J_{\Lambda P}$ **gmres**(J_{PP}, R_P) {Form residual for 2nd Schur complement}
    $\delta\Lambda =$ **gmres**(J_S, R_S) {Solve matrix-free linear interface problem}
    $\delta P =$ **gmres**$(J_{PP}, R_P + J_{P\Lambda}\delta\Lambda)$ {Recover pressure increment}
    $[P^{n,k+1}, \Lambda^{n,k+1}] = [P^{n,k} + \delta P, \Lambda^{n,k} + \delta\Lambda]$ {Increment unknowns}
  end for
  $[P^n, \Lambda^n] = [P^{n,k}, \Lambda^{n,k}]$ {Newton has converged}
end for

the implementation of the corresponding Newton-GMRES–GMRES algorithm in Algorithm 2. Here, a linear interface iteration is performed that requires the solution of linear subdomain problems. The GJS method is also fully parallelizable; both distributed matrix-vector operations and the block diagonal structure of J_{PP} can be utilized for parallelism. We remark that the linear interface iteration is matrix-free, i.e., J_S is not formed, but its action is computed. Therefore, preconditioners for the interface GMRES iteration require techniques such as a balancing preconditioner [29] or the construction of a multiscale flux basis [17].

Remark 2.1. In the single-phase model (2.1a)–(2.1f), the compressibility of the fluid is the cause of both time dependence and nonlinearity. If $c = 0$ in the equation of state (2.2), then the fluid density is constant, and the system becomes a steady state linear problem, so long as $q = q(\boldsymbol{x})$. By arbitrarily specifying a zero initial condition $p_0 = 0$ and normalized time step $\delta t = 1$, the linear problem can be solved in

a single Newton iteration for a single time step, using Algorithm 1 or 2. The solution to the 2nd Schur complement system for the linear problem is the interface iteration of [6, 7].

2.5. Solvability of the Jacobian systems. The three lemmas below justify the use of GMRES in both the GJ and GJS algorithms. We make the following assumptions.

Assumption 2.1. The matrix B is injective in the case $c = 0$.

Assumption 2.2. The matrix L is injective.

The former is true for mixed methods that satisfy an inf-sup condition. Recall that in the case $c = 0$ we assume that functions in the pressure space have mean value zero in Ω, which gives us an inf-sup stable discretization for no-flow boundary conditions. The latter is true when the number of degrees of freedom in the mortar space is not too large in relation to the degrees of freedom on the traces of the subdomain. For a sufficient condition, see [7], Assumption 2.1.

LEMMA 2.1. *The matrix J is positive definite.*

Proof. Observe from (2.8a), (2.3), and (2.9) that A^{-1} is symmetric and positive definite, i.e.,

$$\exists a_0 > 0 : \quad (\delta U)^T A^{-1} \delta U \geq a_0 \|\delta U\|^2 \quad \forall \delta U \in \mathbb{R}^{N_{\boldsymbol{u}}}.$$

Similarly, (2.8e) and (2.9) imply that D is symmetric negative definite when $c > 0$, i.e.,

$$\exists d_0 > 0 : \quad -(\delta P)^T D \delta P \geq d_0 \|\delta P\|^2 \quad \forall \delta P \in \mathbb{R}^{N_p}.$$

If $c = 0$, then $D = 0$ and the above inequality holds with $d_0 = 0$. We symmetrize J by a similarity transformation with the simple diagonal scaling

$$\begin{aligned} &\begin{bmatrix} \sqrt{\delta t}\, I & 0 \\ 0 & I \end{bmatrix}^{-1} J \begin{bmatrix} \sqrt{\delta t}\, I & 0 \\ 0 & I \end{bmatrix} \\ &\quad = \begin{bmatrix} \delta t\, B^T A^{-1} B - D & \sqrt{\delta t}\, B^T A^{-1} L \\ \sqrt{\delta t}\, (B^T A^{-1} L)^T & L^T A^{-1} L \end{bmatrix} = J^{\mathrm{sym}}. \end{aligned} \tag{2.13}$$

Letting $\xi^T = [(\delta P)^T \ (\delta \Lambda)^T]^T$, we have

$$\begin{aligned} \xi^T J^{\mathrm{sym}} \xi &= (\sqrt{\delta t}\ B\ \delta P + L\ \delta \Lambda)^T A^{-1} (\sqrt{\delta t}\ B\ \delta P + L\ \delta \Lambda) - (\delta P)^T D\ \delta P \\ &\geq a_0 \|\sqrt{\delta t}\ B\ \delta P + L\ \delta \Lambda\|^2 + d_0 \|\delta P\|^2 \geq 0. \end{aligned}$$

Consider the case $c > 0$. If $\xi^T J^{\mathrm{sym}} \xi = 0$, then $\sqrt{\delta t}\ B\ \delta P + L\ \delta \Lambda = 0$ and $\delta P = 0$. Since L is injective, this implies that $\delta \Lambda = 0$. The case $c = 0$ is handled by an argument similar to Lemma 2.1 in [6]. ∎

LEMMA 2.2. *The matrix J_{PP} is positive definite.*

Proof. We have, $\forall \delta P \in \mathbb{R}^{N_p}$,

$$\begin{aligned} (\delta P)^T J_{PP}\ \delta P &= \delta t\ (B\ \delta P)^T A^{-1} (B\ \delta P) - (\delta P)^T D\ \delta P \\ &\geq \delta t\ a_0 \|B\ \delta P\|^2 + d_0 \|\delta P\|^2 \geq 0. \end{aligned}$$

Assume that $(\delta P)^T J_{PP}\ \delta P = 0$. When $c > 0$, then $d_0 > 0$, and it follows that $\delta P = 0$. When $c = 0$, $B\ \delta P = 0$ implies that $\delta P = 0$, since B is injective. ∎

LEMMA 2.3. *The matrix J_S is positive definite.*

Proof. Observe that matrix J^{sym} in (2.13) has the same diagonal blocks as matrix J. By direct calculation, it is also true that the Schur complement J_S^{sym} is equal to

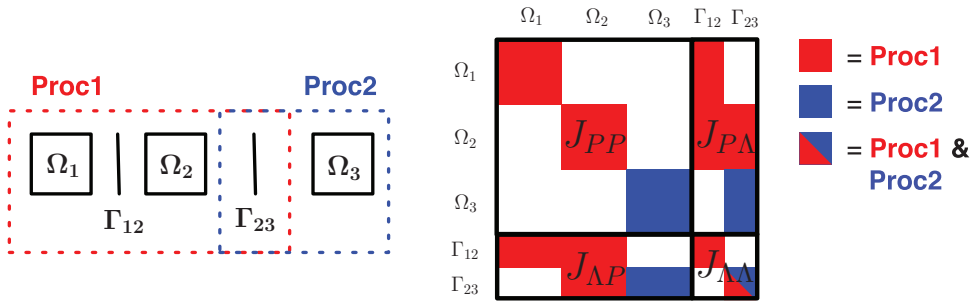

FIG. 3. *Illustration of processor ownership. Left shows computational discretization and right shows storage of the GJ matrix for the* 1*st Schur complement.*

the Schur complement J_S in (2.12). By Lemma 2.1 and Theorem 7.7.6 in [21], J_S is positive definite. ∎

2.6. Implementation of parallel solvers and preconditioners. It is oftentimes advantageous to implement codes where processors share ownership of subdomains and mortars. Figure 3 shows a simple scenario with three subdomains and two mortars.

In the GJ method, the global multiscale system (2.11) is explicitly formed. In this case any generic, high performance solver library implementing parallel iterative linear solver and parallel preconditioner can be utilized. The GJS method requires the solutions to linear, matrix-free interface problems. Therefore a parallel iterative solver library is needed that specifically allows the user to specify the action of the interface operator in the code itself. In recent years, several matrix-free preconditioners have successfully been developed for linear interface problems, such as the balancing preconditioner [29] and the construction of a multiscale flux basis [17]. On the other hand, preconditioners for nonlinear interface problems, as were needed in the FD method, were difficult to formulate and implement [42, 16]. The advantage of the GJS method is that linear interface preconditioners can now be applied to the nonlinear problem. In this work, we choose to report only unpreconditioned results with the GJS method. The reason is that our goal is focused on introducing the GJ and GJS methods, and to emphasize their simplicity over the FD method. It is impossible to use the same preconditioning strategy for both GJ and GJS methods, since they are solving fundamentally different linear systems. However with a large but relatively simple problem, it is possible to directly compare GJ and GJS without preconditioning.

3. Comparison to the forward difference method. Recalling Figure 1, we describe the previous algorithm [38, 42, 30, 28, 16] for solving the MMMFEM with nonlinear model problems, i.e., the FD method. Both the FD and GJ methods seek to solve the same coupled system of nonlinear equations (2.4a)–(2.4c). Therefore, the first two steps are the same as in section 2, leading to the nonlinear algebraic system (2.7a)–(2.7c). The remainder consists of the following:

3. Formulate a nonlinear interface problem and linearize it with respect to mortar variable only, using an FD approximation of the interface Jacobian;
4. linearize the subdomain problems with respect to velocity and pressure unknowns, with fixed interface data;
5. eliminate subdomain velocities.

To formulate the FD method, using (2.7a)–(2.7b), we consider the subdomain pressure coefficients P and velocity coefficients U as implicit functions of the interface data Λ, that is,

$$U = U(\Lambda) \quad \text{and} \quad P = P(\Lambda). \tag{3.1}$$

With these, the formal definition of the interface residual (2.6c) as a function of Λ is $Z : \mathbb{R}^{N_\lambda} \to \mathbb{R}^{N_\lambda}$ such that

$$Z(\Lambda) = H(U(\Lambda)). \tag{3.2}$$

Using (2.7c), this reduces the global problem (2.7a)–(2.7c) on time step n to finding Λ such that

$$Z(\Lambda) = 0. \tag{3.3}$$

Since this nonlinear interface problem implicitly requires the solution to nonlinear subdomain problems (2.7a)–(2.7b), the action of the interface Jacobian cannot be computed explicitly. Therefore, an FD approximation is used.

Let $J_Z(\Lambda)\mu$ denote the Jacobian of Z computed at Λ in the direction of $\mu \in \mathbb{R}^{N_\lambda}$. For a small step size $\alpha \approx 10^{-6}$, let $\tilde{\alpha} = \alpha\|\Lambda\|/\|\mu\|$ be a normalized step size. The FD approximation is given by

$$J_Z(\Lambda)\mu \approx J_{Z,\alpha}(\Lambda)\mu = \begin{cases} 0, & \mu = 0, \\ \frac{Z(\Lambda+\tilde{\alpha}\mu)-Z(\Lambda)}{\tilde{\alpha}}, & \mu \neq 0, \Lambda \neq 0, \\ \frac{Z(\alpha\mu/\|\mu\|)-Z(0)}{\alpha/\|\mu\|}, & \mu \neq 0, \Lambda = 0. \end{cases} \tag{3.4}$$

This approximation renders the inexact Newton's method [27] for solving (3.3). The linearized interface problem at each inexact Newton step becomes

$$J_{Z,\alpha}(\Lambda)\delta\Lambda = -Z(\Lambda) = -R_Z. \tag{3.5}$$

The solution to this system is obtained with a matrix-free GMRES iteration, as described in Algorithm 3.

On the subdomain level, the implicit functions (3.1) become residual equations (2.7a)–(2.7b) with fixed interface data Λ, giving the linearized system

$$\begin{bmatrix} A & B \\ \delta t\ B^T & D \end{bmatrix} \begin{bmatrix} \delta U \\ \delta P \end{bmatrix} = - \begin{bmatrix} F \\ G \end{bmatrix}. \tag{3.6}$$

Notice the similarity of system (3.6) in comparison to system (2.10). This means a code that implements the FD method can be modified to solve the GJ or GJS methods with only modest changes. The final step of eliminating subdomain velocities gives

$$\begin{aligned} &\left(\delta t\ B^T A^{-1} B - D\right) \delta P = G - \delta t\ B^T A^{-1} F \\ \iff \quad &J_{PP}\, \delta P = R_P. \end{aligned} \tag{3.7}$$

ALGORITHM 3. THE FD METHOD.

{Superscript n denotes time index and n, k Newton index k at time n.}
Given $[P^0, \Lambda^0]$, {Initial condition}
for $n = 1, \ldots, N_T$ {Time index} **do**
  $\Lambda^{n,0} = \Lambda^{n-1}$ {Initial guess for Newton}
  for $k = 0, \ldots,$ NEWT_MAX {Newton index} **do**
    $[R_Z^{n,k}, P^{n,k}] = Z(\Lambda^{n,k})$
      {Form residual and store associated pressure; **requires subdomain solve**}
    if ($\|R_Z^{n,k}\| <$ NEWT_TOL) **then break** k-loop {Check nonlinear convergence}
    function $J_{Z,\alpha}(\Lambda^{n,k})\xi$ {**return** $\tilde{\alpha}^{-1}(Z(\Lambda^{n,k} + \tilde{\alpha}\xi) - R_Z^{n,k})$}
      {Define FD approximation to J_Z; **application requires subdomain solve**}
    $\delta\Lambda =$ **gmres**$(J_{Z,\alpha}, -R_Z^{n,k})$ {Solve matrix-free linear interface problem}
    $\Lambda^{n,k+1} = \Lambda^{n,k} + \delta\Lambda$ {Increment unknown}
  end for
  $[P^n, \Lambda^n] = [P^{n,k}, \Lambda^{n,k}]$ {Newton has converged}
end for

As observed in section 2, the subdomain Jacobian J_{PP} has block diagonal structure with each decoupled block corresponding to one subdomain. Thus each subdomain problem can be solved separately in parallel. Once the subdomain solution (U, P) is known, the flux jump for the given Λ is computed with (3.3). The subdomain solve is given in Algorithm 4.

ALGORITHM 4. NONLINEAR SUBDOMAIN SOLVE $Z : \mathbb{R}^{N_\lambda} \to \mathbb{R}^{N_\lambda}$.

{Superscript k denotes the Newton index.}
Given Λ, {Fixed interface data}
Given P^0, {Initial guess for subdomain Newton}
for $k = 0, \ldots,$ NEWT_MAX {Newton index} **do**
  $R_P^k(P^k; \Lambda)$ {Form residual}
  if ($\|R_P^k\| <$ NEWT_TOL) **then break** k-loop {Check nonlinear convergence}
  $J_{PP}^k(P^k; \Lambda)$ {Form subdomain Jacobian}
  $\delta P =$ **gmres**(J_{PP}, R_P) {Solve subdomain system}
  $P^{k+1} = P^k + \delta P$ {Increment unknown}
end for
return $Z(\Lambda)$ using definition (3.2) {Return the corresponding jump in flux}

Several remarks are warranted about the FD method. First, as described in [27], the approximation (3.4) is nonlinear with respect to the search direction μ, due to normalization in the denominator. This is true even if the original model problem is linear. Therefore, the application of the FD method to a linear model may result in more than one interface Newton step. Second, each time the action of $J_{Z,\alpha}$ is needed, it requires the action of nonlinear subdomain operators Z. These nested Newton iterations make the FD method quite costly in comparison to methods with a single Newton iteration, such as the GJ and GJS methods, the enhanced velocity method [37], and single domain, multiple processor data decomposition [36]. Third, it is not possible to analytically form an $N_\lambda \times N_\lambda$ matrix to solve (3.5), since it implicitly requires the action of nonlinear subdomain operators. This means that

specialized nonlinear preconditioners, such as [16], were needed for the FD method, whereas the GJS method can use linear preconditioning techniques. This is also the reason why a generic linear solver can only be used on the subdomain level for the FD method, unlike the GJ method, where such solvers can be applied to the global system. It should be noted that since both FD and GJS involve solving an interface problem with subdomain problems solved at each interface iteration, they have reduced interprocessor communication and exhibit improved parallel scalability.

4. Numerical results. In this section, we perform four sets of numerical tests: Example 4.1 is a small linear verification test, Example 4.2 is a large nonlinear homogeneous test without preconditioning, Example 4.3 is a numerical study on condition number, and Example 4.4 is a large nonlinear heterogeneous test with preconditioning. All tests were performed on the TACC Lonestar supercomputer [12] using research codes developed at the University of Texas at Austin. Our convention is that the x-coordinate represents the vertical dimension. In the following examples, each subdomain is associated with a separate computer processor.

4.1. Small linear verification test. The first example provides a verification test to show that the two new algorithms proposed in this work yield the same numerical solution as the previous implementation of the MMMFEM for a small steady state incompressible problem in two dimensions with no gravity. Since this is a linear problem, in lieu of an FD method, the conventional approach is to use the superposition principle to separate the homogeneous and nonhomogeneous pieces of the interface problem; see, for example, [7]. In this context, the three methods are the

1. GJ method (Algorithm 1), global linear problem without an interface iteration;
2. GJS method (Algorithm 2), linear interface iteration with linear subdomain problems;
3. MMMFEM implementation using superposition.

The domain is the unit square $\Omega = (0,1)^2$, split evenly into two subdomains at $y = 0.5$. Starting with a smooth, nontrivial, manufactured pressure solution

$$p = x^2 + x^3y^4 + \cos(y)\sin(xy),$$

and using homogeneous permeability data $K = diag(1,2)$, we analytically obtain the corresponding Darcy velocity $\boldsymbol{u}$ and source function q. Pressure specified boundary conditions are used on $\partial\Omega$. The relative tolerance for all linear solvers is taken to be 1E-6. Both codes are run with dimensionless units.

All three methods are verified to produce identical results for numerical error and convergence rates in both pressure and velocity unknowns, as shown in Table 1 for three grid configurations. The discrete L^2 norm $|||\cdot|||$ is defined by the midpoint rule evaluated at cell centers. Moreover, these numerical results agree with the theoretical rates proven in [7].

4.2. Large nonlinear unpreconditioned test. The second example has a three-dimensional (3D) domain Ω of size $1000 \times 2000 \times 2000$ (ft), uniformly divided into $100 \times 100 \times 100 = 1$ million elements. This problem is nonlinear with a fluid compressibility of $c = 4.0 \times 10^{-5}$ (1/psi), and has gravitational acceleration. There is a quarter five-spot well pattern, meaning a source of 520 (psi) and a sink of 480 (psi) bottom hole pressure in opposite corners, no-flow external boundary conditions, and a hydrostatic initial condition of $p(0) = 500$ (psi) at the top of the reservoir.

TABLE 1
Convergence results for Example 4.1 *with three grid configurations: matching subdomain grids with matching grid constant mortar (top), nonmatching subdomain grids with continuous linear mortar (middle), and nonmatching subdomain grids with discontinuous quadratic mortar (bottom).*

Ω_1 grid	Γ_{12} grid	Ω_2 grid	$\|\|\|p - p_h\|\|\|$	Rate	$\|\|\|\mathbf{u} - \mathbf{u}_h\|\|\|$	Rate
10×5	10 P0	10×5	1.92E-3	–	1.24E-2	–
$2 \times x10$	20 P0	20×10	4.86E-4	$\mathcal{O}(h^{1.98})$	3.40E-3	$\mathcal{O}(h^{1.87})$
40×20	40 P0	40×20	1.22E-4	$\mathcal{O}(h^{1.99})$	9.16E-4	$\mathcal{O}(h^{1.89})$
80×40	80 P0	80×40	3.05E-5	$\mathcal{O}(h^{2.00})$	2.43E-4	$\mathcal{O}(h^{1.91})$
160×80	160 P0	160×80	7.62E-6	$\mathcal{O}(h^{2.00})$	6.44E-5	$\mathcal{O}(h^{1.91})$

Ω_1 grid	Γ_{12} grid	Ω_2 grid	$\|\|\|p - p_h\|\|\|$	Rate	$\|\|\|\mathbf{u} - \mathbf{u}_h\|\|\|$	Rate
10×5	5 P1	12×6	1.44E-3	–	1.60E-2	–
20×10	10 P1	24×12	3.65E-4	$\mathcal{O}(h^{1.98})$	5.13E-3	$\mathcal{O}(h^{1.64})$
40×20	20 P1	48×24	9.15E-5	$\mathcal{O}(h^{2.00})$	1.68E-3	$\mathcal{O}(h^{1.60})$
80×40	40 P1	96×48	2.29E-5	$\mathcal{O}(h^{2.00})$	5.66E-4	$\mathcal{O}(h^{1.57})$
160×80	80 P1	192×96	5.73E-6	$\mathcal{O}(h^{2.00})$	1.94E-4	$\mathcal{O}(h^{1.55})$

Ω_1 grid	Γ_{12} grid	Ω_2 grid	$\|\|\|p - p_h\|\|\|$	Rate	$\|\|\|\mathbf{u} - \mathbf{u}_h\|\|\|$	Rate
10×5	2 P2dc	12×6	1.44E-3	–	1.08E-2	–
20×10	5 P2dc	24×12	3.65E-4	$\mathcal{O}(h^{1.98})$	3.15E-3	$\mathcal{O}(h^{1.78})$
40×20	10 P2dc	48×24	9.15E-5	$\mathcal{O}(h^{2.00})$	8.96E-4	$\mathcal{O}(h^{1.82})$
80×40	20 P2dc	96×48	2.29E-5	$\mathcal{O}(h^{2.00})$	2.58E-4	$\mathcal{O}(h^{1.79})$
160×80	40 P2dc	192×96	5.74E-6	$\mathcal{O}(h^{2.00})$	7.70E-5	$\mathcal{O}(h^{1.74})$

TABLE 2
Grid configurations for Example 4.2.

Subdomains (processors)	DOFs per subdomain	Total mortar DOFs for P0 mortar	Total mortar DOFs for P1 mortar
$1 \times 2 \times 1 = 2$	500,000	10,000	2,601
$1 \times 2 \times 2 = 4$	250,000	20,000	5,304
$1 \times 4 \times 2 = 8$	125,000	40,000	10,608
$1 \times 4 \times 4 = 16$	62,500	60,000	15,912
$1 \times 8 \times 4 = 32$	31,250	100,000	27,132
$1 \times 8 \times 8 = 64$	15,625	140,000	39,984
$1 \times 16 \times 8 = 128$	7,812	220,000	65,688
$1 \times 16 \times 16 = 256$	3,906	300,000	97,920

The simulation time is $T = 61.2$ (days) with a time step of $\delta t = 0.1$ (days). The permeability is $K = diag(100, 200, 200)$ (md), porosity is $\phi = 0.2$, and fluid viscosity is $\mu = 2.0$ (cp).

The purpose of this example is to compare the unpreconditioned parallel scaling of the FD, GJ, and GJS methods. We choose two types of mortar grid configurations: a P0 mortar grid that matches the traces of subdomain grids, and nonmatching P1 mortar grid with $H = 2h$. We vary the number of subdomains from 2 to 128. Table 2 summarizes these configurations.

Mortars can be useful in both the cases where subdomain grids are either matching or nonmatching on the interfaces, by controlling the strength of flux continuity between subdomains. When subdomain grids are matching, a matching grid mortar can allow direct comparison between single domain, multiple processor (SDMP), GJ, GJS, and FD methods, and a nonmatching grid mortar can create an easier interface problem with fewer degrees of freedom. When subdomain grids are nonmatching, the use of a nonmatching mortar of linear or higher degree makes it possible to achieve multiscale resolution.

TABLE 3
Tolerances used in Example 4.2.

SDMP		GJ		GJS		FD	
Global Newton	1E-6	Global Newton	1E-6	Global Newton	1E-6	Forward difference	1E-6
Global GMRES	1E-6	Global GMRES	1E-6	Interface GMRES	1E-2	Interface Newton	1E-6
				Subdom. GMRES	1E-6	Interface GMRES	1E-2
						Subdom. Newton	1E-6
						Subdom. GMRES	1E-6

TABLE 4
Iteration and timing statistics for Example 4.2. *From top to bottom: GJ, GJS, and FD Methods with nonmatching grid P*1 *mortar.*

P1 Mortar - GJ Method

Number of subdomains (processors)	2	4	8	16	32	64	128
Avg. global Newton iter./time step	1.01	1.01	1.01	1.01	1.01	1.01	1.01
Avg. global GMRES iter./time step	252	252	253	253	253	253	253
Percent communication time	1.4	1.2	65.6	42.7	72.6	69.5	83.1
Total runtime in seconds	6,265	5,524	3,385	1,553	824	435	240

P1 Mortar - GJS Method

Number of subdomains (processors)	2	4	8	16	32	64	128
Avg. global Newton iter./time step	1.01	1.01	1.01	1.01	1.01	1.01	1.01
Avg. intf. GMRES iter./time step	4.31	5.63	6.51	6.51	7.56	8.53	9.74
Avg. subdom. GMRES iter./time step	735	874	960	941	983	923	900
Percent communication time	0.2	0.9	57.8	33.6	69.2	60.1	59.9
Total runtime in seconds	19,656	17,614	9,551	4,114	1,790	656	180

P1 Mortar - FD Method

Number of subdomains (processors)	2	4	8	16	32	64	128
Avg. intf. Newton iter./time step		5.20	5.28	5.42	5.71	5.75	6.21
Avg. intf. GMRES iter./time step		32	33	34	41	45	58
Avg. subdom. Newton iter./time step		65	66	69	82	90	115
Avg. subdom. GMRES iter./time step		3,287	3,329	3,423	3,673	3,680	3,886
Total runtime in seconds	>24 hrs.	79,576	50,181	21,541	12,131	6,107	2,790

In the matching grid example, we also compare runtimes to the SDMP data decomposition method. The tolerances used in the linear and nonlinear iterations for each method are summarized in Table 3. Note that the interface GMRES tolerances for the FD and GJS methods were relaxed, so that the subproblems on inner iterations are not oversolved. Although the structure of the algorithms is quite different, the outermost Newton tolerances are equal in all four methods so that the comparison is fair.

The raw data on the average iteration counts and parallel runtimes for the various methods are given in Table 4. These runtimes are also plotted in Figure 4 for both matching and nonmatching cases. The logarithmic axes of these plots reveal the significant absolute difference in the runtimes. In all cases, the FD method is by far the most expensive, due to complex nested iterations as described in section 3. With the FD method, data are not available for 2 and 4 processors on matching grid, and this simulation took longer than 24 hours. The runtimes of the GJ and GJS methods are much less than the FD method, while all three methods obtain the same solution.

Since the problem size is large enough, all of the methods have parallel scaling up to the largest number of processors run. For a low number of processors, the GJS method is more expensive than the GJ method. However, the slope of the GJS curve shows it scales extremely well in parallel, becoming less expensive than the GJ method above 64 processors. To explain these effects, we report percentage of total

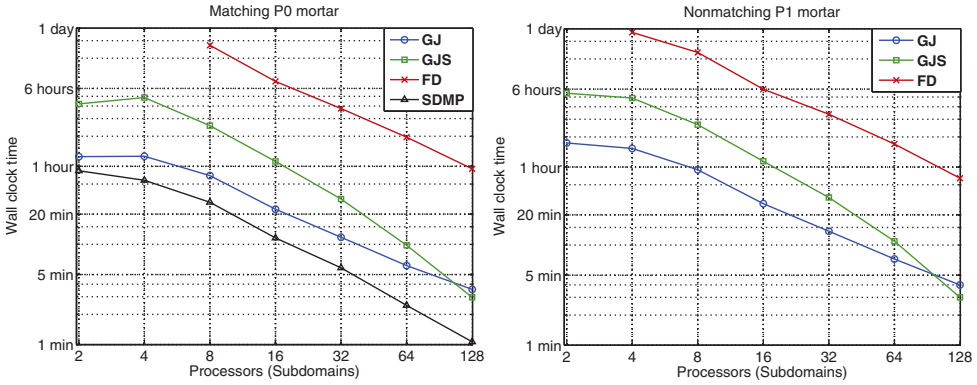

FIG. 4. *Unpreconditioned parallel runtimes versus number of processors in Example* 4.2 *with matching* $P0$ *mortar (left) and nonmatching* $P1$ *mortar (right).*

runtime spent on interprocessor communication in Table 4 for both the GJ and GJS methods. The GJ method requires global communication in its linear iteration, while the GJS method requires only local communication between neighboring subdomains in the interface iteration. As the number of subdomains increases, this allows the GJS method to eventually become less costly than the GJ method. It is important to note that for each method, as the number of subdomains is increased, parallel scalability will stop when the reduction in computation time is dominated by the requisite increase in the communication time. We note that this will be highly machine and problem dependent.

In the matching grid case, the runtimes of the GJ and GJS methods are also compared to the SDMP method. While the SDMP method is faster in this example, the mortars still have an advantage. They allow solving the problem as a GJS interface problem that has better parallel scaling due to the reduced communications cost. However, the most important advantage of the mortars is that they allow using nonmatching grids. These grids have the capability of producing multiscale solutions and they can be adaptively refined, e.g., around wells or channels. For an example of a previous study including parallel scalability of the mortar method with a nonlinear model, see [28].

Table 4 also shows several other trends. For the GJ method, the number of global GMRES iterations does not significantly increase, although the size of the linear system slightly increases, as the number of mortar variables grows. This gives evidence that the condition number of the global linear systems increases very modestly when adding more subdomains to a problem of a fixed size (see Example 4.3). The average number of subdomain GMRES iterations for the GJS and FD methods gets smaller (per outer iteration) as the size of the subdomains gets smaller. However, the cumulative number of these iterations per time step increases with the number of subdomains, because of the nested loops in these algorithms. The GJ and GJS methods take approximately one global Newton step per time step because there are no approximations in the Jacobian for a slightly nonlinear model. These iterations are far less complex than the FD method, which is seen to take 5–6 interface Newton steps and 65–120 subdomain Newton steps per time step.

4.3. Numerical study on condition number. In this example we study numerically the condition number of the GJ method. The same model parameters as Example 4.2 are used, with a fixed fine grid of $2 \times 100 \times 100 = 20{,}000$ elements.

TABLE 5

Condition number estimates for J matrix for Example 4.3*, with matching grid P*0 *mortar and nonmatching grid P*1 *mortar.*

	P0 mortar		P1 mortar	
Subdomains	Total DOF	Condition number	Total DOF	Condition number
$1 \times 1 \times 1 = 1$	20000	2.37E+2	20000	2.37E+2
$1 \times 2 \times 1 = 2$	20200	8.69E+2	20102	1.73E+3
$1 \times 2 \times 2 = 4$	20400	1.03E+3	20208	1.73E+3
$1 \times 4 \times 2 = 8$	20800	1.03E+3	20416	1.81E+3
$1 \times 4 \times 4 = 16$	21200	1.03E+3	20624	1.81E+3
$1 \times 8 \times 4 = 32$	22000	1.03E+3	21064	1.89E+3
$1 \times 8 \times 8 = 64$	22800	1.04E+3	21568	1.89E+3

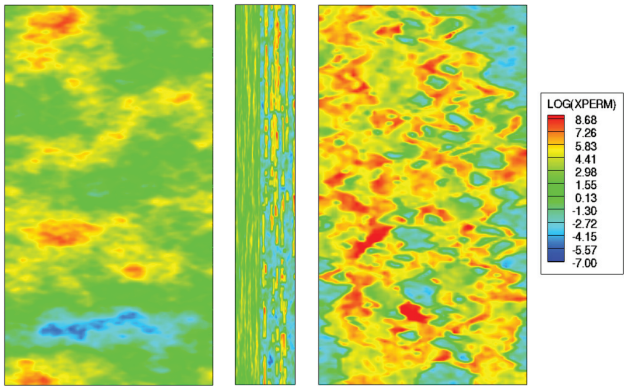

FIG. 5. $\log x$ *permeability (md) for Example* 4.4*. Left shows top layer in the y-z plane, middle shows a side in the x-z plane, and right shows bottom layer in the y-z plane.*

On the first Newton step of the first time step, the matrix J in (2.11) is passed into MATLAB, and the 1-norm condition number is estimated using the built-in command `condest`. In Table 5, we report both matching grid P0 mortar and nonmatching grid P1 mortar cases. The table shows that the condition number grows very modestly as the problem is broken into more subdomains and more Lagrange multipliers are added on these interfaces.

4.4. Large nonlinear preconditioned test. The fourth example uses the highly heterogeneous permeability and porosity fields from the SPE10 benchmark problem [14]. Figure 5 shows that the $\log x$ permeability varies over 15 orders of magnitude; the top 50 layers are fluvial and the bottom 35 layers are highly channelized. The 3D domain Ω has size $170 \times 1200 \times 2200$ (ft) at a depth of 12000 (ft), with a uniform discretization into $85 \times 60 \times 220 = 1.112$ million elements. This problem is nonlinear with a fluid compressibility of $c = 3.1 \times 10^{-6}$ (1/psi), and has gravitational acceleration. There is a quarter five-spot well pattern, with a source of 6000 (psi) and a sink of 4000 (psi) bottom hole pressure in opposite corners, no-flow external boundary conditions, and a hydrostatic initial condition of $p(0) = 5000$ (psi) at the top of the reservoir. The simulation time is $T = 3.1$ (days) with a time step of $\delta t = 0.1$ (days). The fluid viscosity is $\mu = 3.0$ (cp).

In this example, the GJ method is used with preconditioning, facilitated by the `Trilinos` library with the `AztecOO` solver and `ML` preconditioning packages [20, 18]. Here, GMRES is used as a solver with a restart at 40 iterations, an algebraic multi-

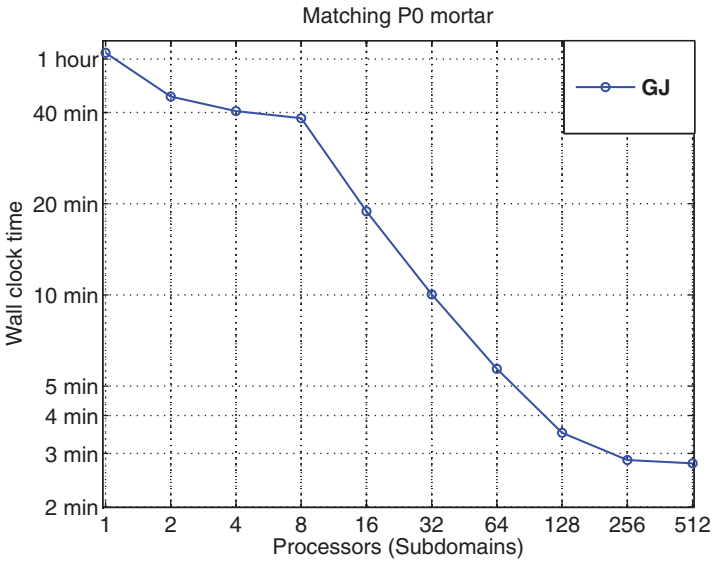

FIG. 6. *Preconditioned parallel runtimes for the GJ method in Example* 4.4 *with matching P*0 *mortar.*

TABLE 6
Grid configurations for Example 4.4.

Subdomains (Processors)	DOFs per subdomain	Total Mortar DOFs for P0 mortar
$1 \times 1 \times 1 = 1$	1,122,000	0
$1 \times 1 \times 2 = 2$	561,000	5,100
$1 \times 1 \times 4 = 4$	280,500	15,300
$1 \times 2 \times 4 = 8$	140,250	34,000
$1 \times 2 \times 8 = 16$	70,125	54,400
$1 \times 4 \times 8 = 32$	35,062	91,800
$1 \times 4 \times 16 = 64$	17,531	132,600
$1 \times 8 \times 32 = 128$	8,765	207,400
$1 \times 16 \times 32 = 256$	4,382	289,000
$1 \times 16 \times 64 = 512$	2,191	438,600

grid V-cycle is used as a preconditioner, ILU(2) is used as a smoother applied with 10 smoothing sweeps, the aggregation operation is unsmoothed, and the coarsest multigrid level is greater than or equal to a 4096×4096 system and is solved with a direct sparse solver. The tolerances used for the GJ method in Example 4.4 are 1E-6 for global Newton and 1E-5 for global GMRES.

Figure 6 shows the preconditioned parallel runtimes for the GJ method with matching P0 mortar. Parallel scalability was obtained up to 512 processors (subdomains), with a decrease in runtime from 62.9 to 2.7 minutes. Grid configurations are summarized in Table 6. The reduction in speedup for a large number of processors is due to very small subdomain size and increased fraction of interprocessor communication. This result demonstrates that the GJ method can be an efficient and scalable way to implement domain decomposition for large heterogeneous nonlinear problems. In the authors' experience, it is very difficult to achieve convergence using the FD method for the SPE10 problem, unless coarser mortars are used. It is also notable that the runtime is shown to decrease from 1 subdomain to 2 subdomains, which is oftentimes not the case for the FD method.

5. Conclusions. We formulated and implemented two GJ algorithms for solving the MMMFEM approximation for slightly compressible, single-phase flow through porous media. These algorithms are arguably more robust than the previous FD algorithm, because they update both mortar and subdomain variables simultaneously in a single Newton iteration. We have implemented the GJ and GJS methods in a parallel high performance computing environment, and we have numerically demonstrated that they outperform the FD method in several large scale example problems. Future work may include formulating the GJ method with more complex flow models including multiphysics problems, and applying linear preconditioning techniques to the GJS method.

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 (2007), 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. Arbogast, M.F. Wheeler, and I. Yotov, *Mixed finite elements for elliptic problems with tensor coefficients as cell-centered finite differences*, SIAM J. Numer. Anal., 34 (1997), pp. 828–852.

[9] D.N. Arnold and F. Brezzi, *Mixed and nonconforming finite element methods: Implementation, postprocessing and error estimates*, RAIRO Model Math. Anal. Numer., 19 (1985), pp. 7–32.

[10] M.T. Balhoff, S.G. Thomas, and M.F. Wheeler, *Mortar coupling and upscaling of pore-scale models*, Comput. Geosci., 12 (2008), pp. 15–27.

[11] M. Benzi, G.H. Golub, and J. Liesen, *Numerical solution of saddle point problems*, Acta Numer., 14 (2005), pp. 1–137.

[12] Texas Advanced Computing Center, http://www.tacc.utexas.edu/.

[13] 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.

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

[15] 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.

[16] B. Ganis, G. Pencheva, M.F. Wheeler, T. Wildey, and I. Yotov, *A frozen Jacobian multiscale mortar preconditioner for nonlinear interface operators*, Multiscale Model. Simul., 10 (2012), pp. 853–873.

[17] 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.

[18] M.W. Gee, C.M. Siefert, J.J. Hu, R.S. Tuminaro, and M.G. Sala, *ML* 5.0 *Smoothed Aggregation User's Guide*, Technical report SAND2006-2649, Sandia National Laboratories, Albuquerque, NM, 2006.

[19] V. Girault, S. Sun, M.F. Wheeler, and I. Yotov, *Coupling discontinuous Galerkin and mixed finite element discretizations using mortar finite elements*, SIAM J. Numer. Anal., 46 (2008), pp. 949–979.

[20] M.A. Heroux and J.M. Willenbring, *Trilinos Users Guide*, Technical report SAND2003-2952, Sandia National Laboratories, Albuquerque, NM, 2003.

[21] R.A. Horn and C.R. Johnson, *Matrix Analysis*, Cambridge University Press, Cambridge, 1990.
[22] 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.
[23] 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.
[24] 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.
[25] 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.
[26] 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.
[27] C.T. Kelley, *Iterative Methods for Linear and Nonlinear Equations*, Frontiers Appl. Math. 16, SIAM, Philadelphia, 1995.
[28] Q. Lu, M. Peszyńska, and M.F. Wheeler, *A parallel multiblock black-oil model in multimodel implementation*, SPE Journal, 7 (2002), pp. 278–287.
[29] G. Pencheva and I. Yotov, *Balancing domain decomposition for mortar mixed finite element methods*, Numer. Linear Algebra Appl., 10 (2003), pp. 159–180.
[30] M. Peszyńska, M.F. Wheeler, and I. Yotov, *Mortar upscaling for multiphase flow in porous media*, Comput. Geosci., 6 (2002), pp. 73–100.
[31] A.M. Quarteroni and A. Valli, *Numerical Approximation of Partial Differential Equations*, Springer Ser. Comput. Math. 23, Springer Verlag, Berlin, 2008.
[32] R.A. Raviart and J.M. Thomas, *A mixed finite element method for 2nd order elliptic problems*, in Mathematical Aspects of the Finite Element Method, Lecture Notes in Math. 606, Springer-Verlag, New York, 1977, pp. 292–315.
[33] Y. Saad and M.H. Schultz, *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Stat. Comput., 7 (1986), pp. 856–869.
[34] A. Toselli and O.B. Widlund, *Domain Decomposition Methods–Algorithms and Theory*, Springer Ser. Comput. Math. 34, Springer Verlag, Berlin, 2005.
[35] D. Vassilev and I. Yotov, *Coupling Stokes–Darcy flow with transport*, SIAM J. Sci. Comput., 31 (2009), pp. 3661–3684.
[36] 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.
[37] J.A. Wheeler, M.F. Wheeler, and I. Yotov, *Enhanced velocity mixed finite element methods for flow in multiblock domains*, Comput. Geosci., 6 (2002), pp. 315–332.
[38] M.F. Wheeler, T. Arbogast, S. Bryant, J. Eaton, Q. Lu, M. Peszynska, and I. Yotov, *A parallel multiblock/multidomain approach for reservoir simulation*, in SPE Reservoir Simulation Symposium, Honolulu, HI, SPE, Dallas, TX, SPE 51884-MS, 1999.
[39] M.F. Wheeler and I. Yotov, *Physical and computational domain decompositions for modeling subsurface flows*, in Domain Decomposition Methods 10, Contemp. Math. 218, 1998, pp. 217–228.
[40] M.F. Wheeler and I. Yotov, *Multigrid on the interface for mortar mixed finite element methods for elliptic problems*, Comput. Methods Appl. Mech. Engrg., 184 (2000), pp. 287–302.
[41] M.F. Wheeler and I. Yotov, *A multipoint flux mixed finite element method*, SIAM J. Numer. Anal., 44 (2006), pp. 2082–2106.
[42] 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.
[43] I. Yotov, *Interface solvers and preconditioners of domain decomposition type for multiphase flow in multiblock porous media*, Adv. Comput. Theory Practice, 7 (2001), pp. 157–167.