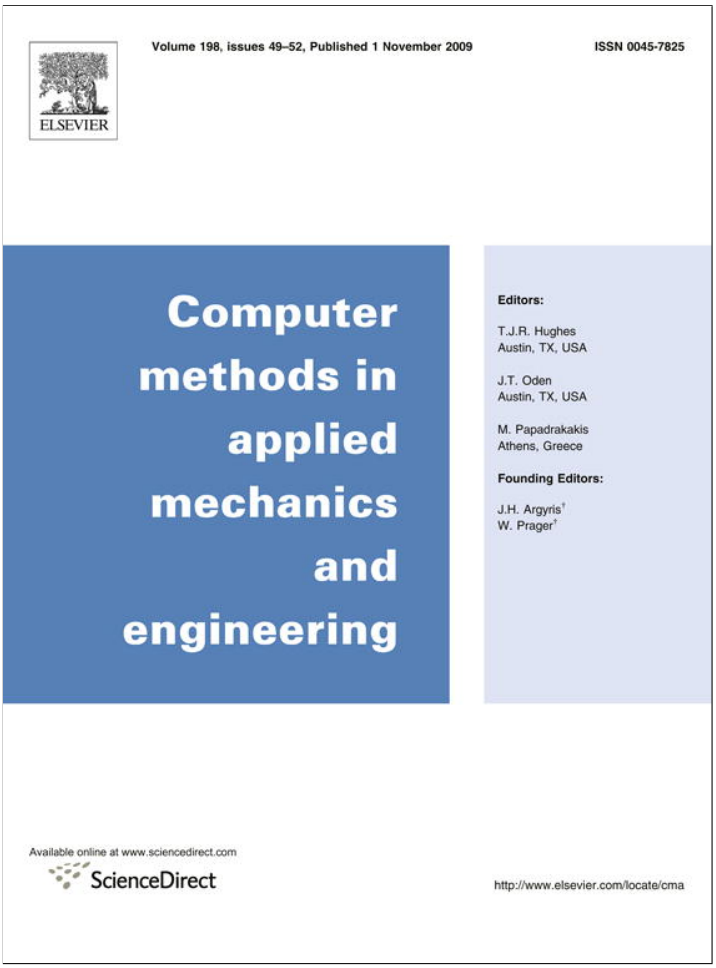
Provided for non-commercial research and education use.
Not for reproduction, distribution or commercial use.



This article appeared in a journal published by Elsevier. The attached copy is furnished to the author for internal non-commercial research and education use, including for instruction at the authors institution and sharing with colleagues.

Other uses, including reproduction and distribution, or selling or licensing copies, or posting to personal, institutional or third party websites are prohibited.

In most cases authors are permitted to post their version of the article (e.g. in Word or Tex form) to their personal website or institutional repository. Authors requiring further information regarding Elsevier's archiving and manuscript policies are encouraged to visit:

http://www.elsevier.com/copyright

Author's personal copy

Comput. Methods Appl. Mech. Engrg. 198 (2009) 3989–3998

Contents lists available at ScienceDirect

Comput. Methods Appl. Mech. Engrg.

journal homepage: www.elsevier.com/locate/cma

Implementation of a mortar mixed finite element method using a Multiscale Flux Basis

Benjamin Ganis *, Ivan Yotov

Department of Mathematics, University of Pittsburgh, 301 Thackeray Hall, Pittsburgh, PA 15260, USA

ARTICLE INFO

Article history:
Received 27 February 2009
Received in revised form 2 September 2009
Accepted 4 September 2009
Available online 11 September 2009

Keywords:
Multiscale
Mortar finite element
Mixed finite element
Porous media flow

ABSTRACT

This paper provides a new implementation of a multiscale mortar mixed finite element method for second order elliptic problems. The algorithm uses non-overlapping domain decomposition to reformulate a fine scale problem as a coarse scale mortar interface problem, which is then solved using an iterative method. The original implementation by Arbogast, Pencheva, Wheeler, and Yotov, Multiscale Model. Simul. 2007, required solving one local Dirichlet problem on each subdomain per interface iteration. We alter this implementation by forming a Multiscale Flux Basis. This basis consists of mortar functions representing the individual flux responses for each mortar degree of freedom, on each subdomain independently. The computation of these basis functions requires solving a fixed number of Dirichlet subdomain problems. Taking linear combinations of the Multiscale Flux Basis functions replaces the need to solve any Dirichlet subdomain problems during the interface iteration. This new implementation yields the same solution as the original implementation, and is computationally more efficient in cases where the number of interface iterations is greater than the number of mortar degrees of freedom per subdomain. The gain in computational efficiency increases with the number of subdomains.

© 2009 Elsevier B.V. All rights reserved.

1. Introduction

This paper provides a new way of implementing the multiscale mortar mixed finite element method (MMMFEM) which was proposed by Arbogast, Pencheva, Wheeler, and Yotov in 2007 [7]. We consider a second order elliptic equation (1), which models single phase flow in porous media. The permeability tensor K varies on a fine scale and so do the velocity $\mathbf{u}$ and the pressure p. Resolving the solution on the fine scale is often computationally infeasible, necessitating multiscale approximations. Our choice of mixed finite element method for the discretization is motivated by its local element-wise conservation of mass and accurate velocity approximation.

The MMMFEM was proposed in [7] as an alternative to existing multiscale methods, such as the variational multiscale method [18,19,4,2,5,3] and multiscale finite elements [16,17,14,11,20,1]. The latter two approaches are closely related [5]. In all three methods 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 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) degrees of freedom.

The variational multiscale method and multiscale finite elements both compute a multiscale basis by solving local fine scale problems with boundary conditions or a source term corresponding to the coarse scale degrees of freedom. This basis is then used to solve the coarse scale problem. The MMMFEM uses a non-overlapping domain decomposition algorithm which introduces a Lagrange multiplier space on the subdomain interfaces to weakly impose certain continuity conditions. By eliminating the subdomain unknowns the global fine scale problem is reduced to an interface problem, which is solved using an iterative method. The domain decomposition algorithm was originally developed for the case of matching grids [15] and then extended to the case of non-matching grids using mortar finite elements [28,6]. This generalization allows for extremely flexible finite element partitions, as both the fine scale elements across subdomain interfaces and the subdomains themselves (i.e. the coarse grid) may be spatially non-conforming. Moreover, one has the ability to vary the interface degrees of freedom [26,23,7]. If only a single mortar grid element is used per interface, the resulting approximation is comparable to the one in the variational multiscale method or multiscale finite elements. In the MMMFEM framework, a posteriori error estimators [27] can be employed to adaptively refine the mortar grids where necessary to improve the global accuracy. Furthermore, higher order mortar approximation can be used to compensate

* Corresponding author.
E-mail addresses: bag8@pitt.edu (B. Ganis), yotov@math.pitt.edu (I. Yotov).

0045-7825/$ - see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.cma.2009.09.009

Author's personal copy

for the coarseness of the mortar grid and obtain fine scale convergence of the error [7]. Thus, the MMMFEM is more flexible than the variational multiscale method and multiscale finite elements. Another observation is that the MMMFEM resolves the flux through the coarse interfaces on the fine scale, which is not the case for the other two approaches.

The original implementation of the MMMFEM in [7] requires solving one Dirichlet fine scale subdomain problem per interface iteration. As a result the number of subdomain solves increases with the dimension of the coarse space, making it difficult to compare the computational efficiency of the method to other existing multiscale methods. In this paper we alter this implementation by forming what we call a Multiscale Flux Basis, before the interface iteration begins. This basis consists of mortar functions representing the individual flux responses from each mortar degree of freedom, on each subdomain independently. These basis functions may also be described as traces of the discrete Green's functions corresponding to the mortar degrees of freedom along the subdomain interfaces. The computation of these basis functions requires solving a fixed number of Dirichlet subdomain problems. Taking linear combinations of the Multiscale Flux Basis functions replaces the need to solve any Dirichlet subdomain problems during the interface iteration. This new implementation yields the same solution as the original implementation and makes the MMMFEM comparable to the variational multiscale method and multiscale finite elements in terms of computational efficiency. In our numerical experiments we compare the computational cost of the new implementation to the one for the original implementation with and without preconditioning of the interface problem. If no preconditioning is used, the Multiscale Flux Basis implementation is computationally more efficient in cases where the number of mortar degrees of freedom per subdomain is less than the number of interface iterations. If balancing preconditioning is used [13,22], the number of iterations is reduced, but each interface iteration requires three subdomain solves. In this case the Multiscale Flux Basis implementation is more efficient if the number of mortar degrees of freedom per subdomain is less than three times the number of interface iterations.

The format of the paper is as follows. Section 2 introduces the MMMFEM method and its step-by-step formulation leading to its original implementation. Section 3 describes our new implementation. In particular, it introduces the concept of a Multiscale Flux Basis, explains how it is used in the interface iteration, and discusses specific implementation details. Section 4 provides several numerical examples which illustrate the computational efficiency of the Multiscale Flux Basis implementation. Section 5 contains concluding remarks and directions for further work.

2. The MMMFEM

Our model problem is a second order linear elliptic equation written as a first order system in mixed form, arising in applications to single phase incompressible flow in porous media. The pressure p and the Darcy velocity $\mathbf{u}$ satisfy the system

$$\alpha p + \nabla \cdot \mathbf{u} = f \quad \text{in } \Omega, \tag{1a}$$
$$\mathbf{u} = -K\nabla p \quad \text{in } \Omega, \tag{1b}$$
$$p = g_D \quad \text{on } \Gamma_D, \tag{1c}$$
$$\mathbf{u}\cdot\mathbf{n} = g_N \quad \text{on } \Gamma_N. \tag{1d}$$

Here Ω is a bounded domain in $\mathbb{R}^d (d = 2 \text{ or } 3)$ with boundary $\partial\Omega = \overline{\Gamma}_D \cup \overline{\Gamma}_N, \Gamma_D \cap \Gamma_N = \emptyset$, and outer unit normal $\mathbf{n}$, $\alpha(x) \geqslant 0$, and $K(\mathbf{x})$ is a symmetric and uniformly positive definite permeability tensor with components in $L^\infty(\Omega)$. We assume that $f \in L^2(\Omega)$, $g_D \in H^{1/2}(\Gamma_D)$, and $g_N \in L^2(\Gamma_N)$.

Throughout the paper, C denotes a generic positive constant independent of the discretization parameters h and H. For a domain $G \subset \mathbb{R}^d$, the $L^2(G)$ inner product and norm for scalar and vector valued functions are denoted $(\cdot,\cdot)_G$ and $\|\cdot\|_G$, respectively. We omit G in the subscript if $G = \Omega$. For a section of the domain or element boundary $S \subset \mathbb{R}^{d-1}$ we write $\langle\cdot,\cdot\rangle_S$ and $\|\cdot\|_S$ for the $L^2(S)$ inner product (or duality pairing) and norm, respectively.

2.1. Domain decomposition

The first step in formulating the MMMFEM is to use the domain decomposition approach described in [15] to restrict the model problem into smaller pieces. The domain Ω is divided into non-overlapping subdomains $\Omega_i, i = 1,\ldots,n$, that are allowed to be spatially non-conforming, so we have $\overline{\Omega} = \bigcup_{i=1}^n \overline{\Omega_i}$ and $\Omega_i \cap \Omega_j = \emptyset$ for $i \neq j$. Denote the single interface between subdomains Ω_i and Ω_j by $\Gamma_{ij} = \partial\Omega_i \cap \partial\Omega_j$, all interfaces that touch subdomain Ω_i by $\Gamma_i = \partial\Omega_i \setminus \partial\Omega$, and the union of all interfaces by $\Gamma = \bigcup_{i\neq j}\Gamma_{ij}$. The domain decomposition can be viewed as a coarse grid on Ω.

System (1) holds within each subdomain Ω_i. The pressure and the normal components of the velocity must be continuous across the interfaces. Equivalently, we seek $(\mathbf{u}_i, p_i)$ such that for $i = 1,\ldots,n$,

$$\alpha p_i + \nabla\cdot\mathbf{u}_i = f \quad \text{in } \Omega_i, \tag{2a}$$
$$\mathbf{u}_i = -K\nabla p_i \quad \text{in } \Omega_i, \tag{2b}$$
$$p_i = g_D \quad \text{on } \partial\Omega_i \cap \Gamma_D, \tag{2c}$$
$$\mathbf{u}_i\cdot\mathbf{n} = g_N \quad \text{on } \partial\Omega_i \cap \Gamma_N, \tag{2d}$$
$$p_i = p_j \quad \text{on } \Gamma_{ij}, \quad i \neq j, \tag{2e}$$
$$\mathbf{u}_i\cdot\mathbf{n}_i + \mathbf{u}_j\cdot\mathbf{n}_j = 0 \quad \text{on } \Gamma_{ij}, \quad i \neq j, \tag{2f}$$

where $\mathbf{n}_i$ is the outer unit normal to $\partial\Omega_i$.

2.2. Variational formulation

The weak pressure and velocity spaces for the global problem (1) are

$$\widetilde{W} = L^2(\Omega), \quad \widetilde{\mathbf{V}} = H(\mathrm{div};\Omega), \quad \widetilde{\mathbf{V}}^\gamma = \{\mathbf{v}\in\widetilde{\mathbf{V}} | \mathbf{v}\cdot\mathbf{n} = \gamma \text{ on } \Gamma_N\},$$

where

$$H(\mathrm{div};\Omega) = \{\mathbf{v}\in(L^2(\Omega))^d | \nabla\cdot\mathbf{v}\in L^2(\Omega)\}.$$

The corresponding dual mixed variational formulation is to find $\mathbf{u}\in\widetilde{\mathbf{V}}^{g_N}$ and $p\in\widetilde{W}$ such that

$$(K^{-1}\mathbf{u},\mathbf{v})_\Omega - (p,\nabla\cdot\mathbf{v})_\Omega = -\langle\mathbf{v}\cdot\mathbf{n}, g_D\rangle_{\partial\Gamma_D} \quad \forall\mathbf{v}\in\widetilde{\mathbf{V}}^0, \tag{3a}$$
$$(\alpha p, w)_\Omega + (\nabla\cdot\mathbf{u}, w)_\Omega = (f,w)_\Omega \quad \forall w\in\widetilde{W}. \tag{3b}$$

Note that the Neumann boundary condition is imposed essentially in the space $\widetilde{\mathbf{V}}^{g_N}$, which is different from the weak velocity test space $\widetilde{\mathbf{V}}^0$.

Similarly, define the weak spaces for each subdomain Ω_i by

$$W_i = L^2(\Omega_i), \quad \mathbf{V}_i = H(\mathrm{div};\Omega_i),$$
$$\mathbf{V}_i^\gamma = \{\mathbf{v}\in\mathbf{V}_i | \mathbf{v}\cdot\mathbf{n} = \gamma \text{ on } \partial\Omega_i\cap\Gamma_N\}.$$

The global weak spaces for the domain decomposition problem (2) are

$$W = \bigoplus_{i=1}^n W_i, \quad \mathbf{V}^\gamma = \bigoplus_{i=1}^n \mathbf{V}_i^\gamma.$$

Note that no continuity is imposed across the interfaces for functions in $\mathbf{V}$ and W. On the interfaces Γ we introduce a Lagrange multiplier space that has a physical meaning of pressure and is used to weakly impose continuity of the normal velocities:

Author's personal copy

$$M = \{\mu \in H^{1/2}(\Gamma) | \mu|_{\Gamma_i} \in (\mathbf{V}_i \cdot \mathbf{n}_i)^*,\ i = 1, \ldots, n\},$$

where $(\cdot)^*$ denotes the dual space.

The corresponding mixed variational formulation is to find $\mathbf{u} \in \mathbf{V}^{g_N}, p \in W$, and $\lambda \in M$ such that for $i = 1, \ldots, n$,

$$(K^{-1}\mathbf{u}, \mathbf{v})_{\Omega_i} - (p, \nabla \cdot \mathbf{v})_{\Omega_i} = -\langle \mathbf{v} \cdot \mathbf{n}_i, \lambda \rangle_{\partial\Omega_i \cap \Gamma} - \langle \mathbf{v} \cdot \mathbf{n}_i, g_D \rangle_{\partial\Omega_i \cap \Gamma_D} \quad \forall \mathbf{v} \in \mathbf{V}_i^0, \tag{4a}$$

$$(\alpha p, w)_{\Omega_i} + (\nabla \cdot \mathbf{u}, w)_{\Omega_i} = (f, w)_{\Omega_i} \quad \forall w \in W_i, \tag{4b}$$

$$\sum_{i=1}^{n} \langle \mathbf{u}_i \cdot \mathbf{n}_i, \mu \rangle_{\Gamma_i} = 0 \quad \forall \mu \in M \tag{4c}$$

Since $\mathbf{V}^\gamma \neq \widetilde{\mathbf{V}}^\gamma$, the extra condition (4c) is needed to weakly enforce the flux continuity lost across the interfaces in the domain decomposition.

The following equivalence result is easy to show.

Lemma 2.1. *If the solution $(\mathbf{u}, p)$ to (3) satisfies (1) in a distributional sense, then $(\mathbf{u}, p, p|_\Gamma)$ solves (4). Conversely, if $(\mathbf{u}, p, \lambda)$ solves (4), then $(\mathbf{u}, p)$ solves (3).*

2.3. Discrete formulation

The multiscale approach to the mortar mixed finite element method combines a local fine scale discretization within each subdomain with a global coarse scale discretization across subdomain interfaces.

First, independently partition each subdomain Ω_i into its own local d-dimensional quasi-uniform affine mesh $\mathcal{T}_{h,i}$. The faces (or edges) of these meshes are spatially conforming within each subdomain, but are allowed to be non-conforming along subdomain interfaces. Let the maximal element diameter of this fine mesh be h_i, and let the global characteristic fine scale diameter be $h = \max_{i=1}^n h_i$. Denote the global fine mesh by $\mathcal{T}_h = \bigcup_{i=1}^n \mathcal{T}_{h,i}$. Let $\mathbf{V}_{h,i} \times W_{h,i} \subset \mathbf{V}_i \times W_i$ be a mixed finite element space on the mesh $\mathcal{T}_{h,i}$ such that $\mathbf{V}_{h,i}$ contains piecewise polynomials of degree k and $W_{h,i}$ contains piecewise polynomials of degree l. Examples of such spaces are the RT spaces [25,21], the BDM spaces [10], the BDFM spaces [9], the BDDF spaces [8], or the CD spaces [12]. Globally, the discrete pressure and velocity spaces for this method are $W_h = \oplus_{i=1}^n W_{h,i}$ and $\mathbf{V}_h = \oplus_{i=1}^n \mathbf{V}_{h,i}$.

Second, we partition each subdomain interface $\Gamma_{i,j}$ with a $(d-1)$-dimensional quasi-uniform affine mesh denoted $\mathcal{T}_{H,i,j}$. This mesh will be the mortar space that weakly enforces continuity of normal fluxes for the discrete velocities across the non-matching grids. Let the maximal element diameter of this coarse mesh be $H_{i,j}$, and let the global characteristic coarse scale diameter be $H = \max_{1 \leqslant i < j \leqslant n} H_{i,j}$. Denote the global coarse mesh by $\mathcal{T}_H = \bigcup_{1 \leqslant i < j \leqslant n} \mathcal{T}_{H,i,j}$. Let $M_{H,i,j} \subset L^2(\Gamma_{i,j})$ be the mortar space containing continuous or discontinuous piecewise polynomials of degree m where $m \geqslant k+1$. Globally, the mortar space for this method is $M_H = \oplus_{1 \leqslant i < j \leqslant n} M_{H,i,j}$. Notice that this is a non-conforming approximation, as $M_H \not\subset M$.

With these finite dimensional subspaces, the multiscale mortar mixed finite element approximation of (4) is to find $\mathbf{u}_h \in \mathbf{V}_h^{g_N}$, $p_h \in W_h$, and $\lambda_H \in M_H$ such that for $i = 1, \ldots, n$,

$$(K^{-1}\mathbf{u}_h, \mathbf{v})_{\Omega_i} - (p_h, \nabla \cdot \mathbf{v})_{\Omega_i} = -\langle \mathbf{v} \cdot \mathbf{n}_i, \lambda_H \rangle_{\partial\Omega_i \cap \Gamma} - \langle \mathbf{v} \cdot \mathbf{n}_i, g_D \rangle_{\partial\Omega_i \cap \Gamma_D} \quad \forall \mathbf{v} \in \mathbf{V}_{h,i}^0, \tag{5a}$$

$$(\alpha p_h, w)_{\Omega_i} + (\nabla \cdot \mathbf{u}_h, w)_{\Omega_i} = (f, w)_{\Omega_i} \quad \forall w \in W_{h,i}, \tag{5b}$$

$$\sum_{i=1}^{n} \langle \mathbf{u}_{h,i} \cdot \mathbf{n}_i, \mu \rangle_{\Gamma_i} = 0 \quad \forall \mu \in M_H. \tag{5c}$$

In this formulation the pressure continuity (2e) is modeled via the mortar pressure function λ_H, while the flux continuity (2f) is imposed weakly on the coarse scale via (5c). For the above method to be well posed, the two scales must be chosen such that the mortar space is not too rich compared to the normal traces of the subdomain velocity spaces.

Assumption 2.1. Assume there exists a constant C independent of h and H such that

$$\|\mu\|_{\Gamma_{ij}} \leqslant C(\|\mathcal{Q}_{h,i}\mu\|_{\Gamma_{ij}} + \|\mathcal{Q}_{h,j}\mu\|_{\Gamma_{ij}}), \quad \forall \mu \in M_H, \quad 1 \leqslant i < j \leqslant n, \tag{6}$$

where $\mathcal{Q}_{h,i} : L^2(\Gamma_i) \to \mathbf{V}_{h,i} \cdot \mathbf{n}_i|_{\Gamma_i}$ is the L^2-projection operator from the mortar space onto the normal trace of the velocity space on subdomain (i), i.e. for any $\phi \in L^2(\Gamma_i)$,

$$\langle \phi - \mathcal{Q}_{h,i}\phi, \mathbf{v} \cdot \mathbf{n}_i \rangle_{\Gamma_i} = 0, \quad \forall \mathbf{v} \in \mathbf{V}_{h,i}. \tag{7}$$

This condition can be easily satisfied in practice by restricting the size of H from below (see e.g. [28,6,22]). Under the above assumption, method (5) is solvable, stable, and accurate [7]. The following result has been shown in [7].

Theorem 2.1. *If Assumption 2.1 holds, then method (5) has a unique solution and there exists a positive constant C, independent of h and H, such that*

$$\|\nabla \cdot (\mathbf{u} - \mathbf{u}_h)\| \leqslant C \sum_{i=1}^{n} \|\nabla \cdot \mathbf{u}\|_{r,\Omega_i} h^r, \quad 0 \leqslant r \leqslant l+1, \tag{8}$$

$$\|\mathbf{u} - \mathbf{u}_h\| \leqslant C \sum_{i=1}^{n} \left(\|p\|_{s+1/2,\Omega_i} H^{s-1/2} + \|\mathbf{u}\|_{r,\Omega_i} h^r + \|\mathbf{u}\|_{r+1/2,\Omega_i} h^r H^{1/2} \right), \quad 1 \leqslant r \leqslant k+1,\ 0 < s \leqslant m+1. \tag{9}$$

Furthermore, if the problem on Ω is H^2-regular, then

$$\|\hat{p} - p_h\| \leqslant C \sum_{i=1}^{n} \left(\|p\|_{s+1/2,\Omega_i} H^{s+1/2} + \|\nabla \cdot \mathbf{u}\|_{t,\Omega_i} h^t H + \|\mathbf{u}\|_{r,\Omega_i} h^r H + \|\mathbf{u}\|_{r+1/2,\Omega_i} h^r H^{3/2} \right), \tag{10}$$

$$\|p - p_h\| \leqslant C \sum_{i=1}^{n} \|p\|_{t,\Omega_i} h^t + \|\hat{p} - p_h\|, \tag{11}$$

where $1 \leqslant r \leqslant k+1, 0 < s \leqslant m+1$, and $0 \leqslant t \leqslant l+1$ and $\hat{p}$ is the L^2-projection of p onto W_h.

2.4. Interface formulation

Following [15], we formulate (5) as an interface problem for the mortar pressure. We decompose the solution to (5) into two parts: $\mathbf{u}_h = \mathbf{u}_h^*(\lambda_H) + \overline{\mathbf{u}}_h$ and $p_h = p_h^*(\lambda_H) + \overline{p}_h$. The first component $(\mathbf{u}_h^*, p_h^*) \in \mathbf{V}_h^0 \times W_h$ solves subdomain problems with zero source and boundary conditions, and has λ_H as a Dirichlet boundary condition along Γ, i.e. for $i = 1, \ldots, n$

$$\left(K^{-1}\mathbf{u}_h^*, \mathbf{v}\right)_{\Omega_i} - \left(p_h^*, \nabla \cdot \mathbf{v}\right)_{\Omega_i} = -\langle \mathbf{v} \cdot \mathbf{n}_i, \lambda_H \rangle_{\partial\Omega_i \cap \Gamma} \quad \forall \mathbf{v} \in \mathbf{V}_{h,i}^0, \tag{12a}$$

$$\left(\alpha p_h^*, w\right)_{\Omega_i} + \left(\nabla \cdot \mathbf{u}_h^*, w\right)_{\Omega_i} = 0 \quad \forall w \in W_{h,i}. \tag{12b}$$

The second component $(\overline{\mathbf{u}}_h, \overline{p}_h) \in \mathbf{V}_h^{g_N} \times W_h$ solves subdomain problems with source f, boundary conditions g_D and g_N on $\partial\Omega$, and zero Dirichlet boundary conditions along Γ, i.e. for $i = 1, \ldots, n$

$$(K^{-1}\overline{\mathbf{u}}_h, \mathbf{v})_{\Omega_i} - (\overline{p}_h, \nabla \cdot \mathbf{v})_{\Omega_i} = -\langle \mathbf{v} \cdot \mathbf{n}_i, g_D \rangle_{\partial\Omega_i \cap \Gamma_D} \quad \forall \mathbf{v} \in \mathbf{V}_{h,i}^0, \tag{13a}$$

$$(\alpha \overline{p}_h, w)_{\Omega_i} + (\nabla \cdot \overline{\mathbf{u}}_h, w)_{\Omega_i} = (f, w)_{\Omega_i} \quad \forall w \in W_{h,i}. \tag{13b}$$

Since the sum of (12a) and (12b) and (13a) and (13b) gives (5a) and (5b), all that remains to do is enforce equation (5c). Thus, the variational interface problem is to find $\lambda_H \in M_H$ such that

$$\sum_{i=1}^{n} \langle -\mathbf{u}_{h,i}^*(\lambda_H) \cdot \mathbf{n}_i, \mu \rangle_{\Gamma_i} = \sum_{i=1}^{n} \langle \overline{\mathbf{u}}_{h,i} \cdot \mathbf{n}_i, \mu \rangle_{\Gamma_i}, \quad \forall \mu \in M_H. \tag{14}$$

Author's personal copy

Equivalently, we define bilinear forms $b_{H,i} : L^2(\Gamma_i) \times L^2(\Gamma_i) \to \mathbb{R}$ and $b_H : L^2(\Gamma) \times L^2(\Gamma) \to \mathbb{R}$ and a linear functional $g_H : L^2(\Gamma) \to \mathbb{R}$ by

$$b_{H,i}(\lambda_{H,i}, \mu) = \langle -\mathbf{u}^*_{h,i}(\lambda_{H,i}) \cdot \mathbf{n}_i, \mu \rangle_{\Gamma_i}, \tag{15a}$$

$$b_H(\lambda_H, \mu) = \sum_{i=1}^{n} b_{H,i}(\lambda_{H,i}, \mu), \tag{15b}$$

$$g_H(\mu) = \sum_{i=1}^{n} \langle \overline{\mathbf{u}}_{h,i} \cdot \mathbf{n}_i, \mu \rangle_{\Gamma_i}. \tag{15c}$$

With these definitions, (14) is equivalent to finding $\lambda_H \in M_H$ such that $b_H(\lambda_H, \mu) = g_H(\mu)$, for all $\mu \in M_H$. The distinction is made between bilinear forms (15a) and (15b) because the former measures the total flux across interface Γ_i and requires no interprocessor communication, while the latter measures the total *jump* in flux across the set of all interfaces Γ and hence does require interprocessor communication.

2.5. Interface iteration

It is easy to check that b_H is symmetric and positive semi-definite on $L^2(\Gamma)$. Moreover, it is positive definite on M_H if Assumption 2.1 holds and either $\Gamma_D \neq \emptyset$ or $\alpha > 0$ [6,7]. Therefore we solve the resulting discrete system with a Conjugate Gradient (CG) algorithm. Define linear operators $B_{H,i} : M_{H,i} \to M_{H,i}$ and $B_H : M_H \to M_H$ and a vector $g_H \in M_H$ corresponding to equations (15) by

$$\langle B_{H,i}\lambda_{H,i}, \mu \rangle_{\Gamma_i} = -\langle \mathbf{u}^*_{h,i}(\lambda_{H,i}) \cdot \mathbf{n}_i, \mu \rangle_{\Gamma_i} \quad \forall \mu \in M_{H,i}, \tag{16a}$$

$$B_H\lambda_H = \sum_{i=1}^{n} B_{H,i}\lambda_{H,i}, \tag{16b}$$

$$\langle g_H, \mu \rangle_{\Gamma} = \sum_{i=1}^{n} \langle \overline{\mathbf{u}}_{h,i} \cdot \mathbf{n}_i, \mu \rangle_{\Gamma_i} \quad \forall \mu \in M_H. \tag{16c}$$

Let $\mathcal{Q}^T_{h,i} : \mathbf{V}_{h,i} \cdot \mathbf{n}_i|_{\Gamma_i} \to M_{H,i}$ be the L^2-orthogonal projection from the normal trace of the velocity space onto the mortar space. Note that (16a) and (16c) imply, respectively,

$$B_{H,i} = -\mathcal{Q}^T_{h,i}\mathbf{u}^*_{h,i}(\lambda_{H,i}) \cdot \mathbf{n}_i, \quad g_H = \sum_{i=1}^{n} \mathcal{Q}^T_{h,i}\overline{\mathbf{u}}_{h,i} \cdot \mathbf{n}_i. \tag{17}$$

Using this notation, the interface formulation is to find $\lambda_H \in M_H$ such that $B_H\lambda_H = g_H$. The operator B_H is known as the Steklov–Poincaré operator [24].

Starting from an initial guess, we iterate on the value of λ_H using the CG algorithm. On each CG iteration, we must evaluate the action of B_H on λ_H. This is done with the following steps:

1. Project mortar data onto subdomain boundaries:
$$\lambda_{H,i} \xrightarrow{\mathcal{Q}_{h,i}} \gamma_i.$$
2. Solve the set of subdomain problems (12) with Dirichlet boundary data γ_i.
3. Project the resulting fluxes from Step 2 back onto the mortar space:
$$-\mathbf{u}^*_h(\gamma_i) \cdot \mathbf{n}_i \xrightarrow{\mathcal{Q}^T_{h,i}} -\mathcal{Q}^T_{h,i}\mathbf{u}^*_h(\gamma_i) \cdot \mathbf{n}_i.$$
4. Compute flux jumps across all subdomain interfaces $\Gamma_{i,j}$:
$$B_H\lambda_H = -\sum_{i=1}^{n} \mathcal{Q}^T_{h,i}\mathbf{u}^*_h(\gamma_i) \cdot \mathbf{n}_i.$$

Steps 1–3 evaluate the action of the flux operator $B_{H,i}$ as in (17) and are done by every subdomain in parallel. Step 4 evaluates the action of the jump operator B_H as in (16b) and requires interprocessor communication across every subdomain interface.

3. Multiscale Flux Basis implementation

Observe that the dominant computational cost in each CG iteration is in the evaluation of the flux operator $B_{H,i}$, which requires solving one Dirichlet subdomain problem per subdomain. One way to potentially reduce this computational cost is with the following approach.

> Before the CG algorithm begins, compute and store the flux responses associated with each mortar degree of freedom, on each subdomain independently.

This is what we call the Multiscale Flux Basis. Its assembly requires solving only a *fixed* number of Dirichlet subdomain problems (12), see Fig. 1. After these solves are completed, the action of $B_{H,i}$ is reduced to taking a linear combination of Multiscale Flux Basis functions. Therefore, if the number of CG iterations exceeds the maximum number of mortar degrees of freedom on any subdomain, then the computational cost will be reduced in terms of fewer required subdomain solves, and should yield faster runtime.

3.1. Assembly of the Multiscale Flux Basis

To compute each function in the Multiscale Flux Basis, we shall apply Steps 1–3 from the interface iteration in order to evaluate the action of the operator $B_{H,i}$ on each mortar basis function, on each subdomain independently. Let there be exactly $N_{H,i}$ mortar degrees of freedom on subdomain (i) and define $\left\{\phi^{(k)}_{H,i}\right\}_{k=1}^{N_{H,i}}$ to be the mortar basis functions for $M_{H,i}$. Then for $\lambda_{H,i} \in M_{H,i}$ we may express

$$\lambda_{H,i} = \sum_{k=1}^{N_{H,i}} \lambda^{(k)}_{H,i}\phi^{(k)}_{H,i}.$$

Consider the kth mortar basis function $\phi^{(k)}_{H,i}$. Computing the Multiscale Flux Basis function corresponding to $\phi^{(k)}_{H,i}$ involves the following steps.

1. Project this mortar basis function onto the subdomain boundary:
$$\mathcal{Q}_{h,i}\phi^{(k)}_{H,i} = \gamma^{(k)}_i.$$
2. Solve problem (12) on each subdomain Ω_i with Dirichlet interface condition $\gamma^{(k)}_i$, i.e. find $\mathbf{u}^*_h = \mathbf{u}^*_h(\gamma^{(k)}_i)$ and $p^*_h = p^*_h(\gamma^{(k)}_i)$ such that
$$(K^{-1}\mathbf{u}^*_h, \mathbf{v})_{\Omega_i} - (p^*_h, \nabla \cdot \mathbf{v})_{\Omega_i} = -\left\langle \mathbf{v} \cdot \mathbf{n}_i, \gamma^{(k)}_i \right\rangle_{\partial\Omega_i \cap \Gamma} \quad \forall \mathbf{v} \in \mathbf{V}^0_{h,i},$$
$$(\alpha p^*_h, w)_{\Omega_i} + (\nabla \cdot \mathbf{u}^*_h, w)_{\Omega_i} = 0 \quad \forall w \in W_{h,i}.$$
3. Project the resulting boundary flux back onto the mortar space:
$$\psi^{(k)}_{H,i} = -\mathcal{Q}^T_{h,i}\mathbf{u}^*_h\left(\gamma^{(k)}_i\right) \cdot \mathbf{n}_i.$$

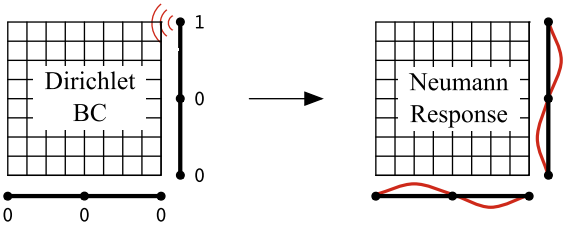


Fig. 1. Illustration of the Multiscale Flux Basis approach.

Author's personal copy

Repeating this procedure for $k = 1, \ldots, N_{H,i}$ forms the Multiscale Flux Basis for subdomain Ω_i:

$$\left\{\psi_{H,i}^{(1)}, \psi_{H,i}^{(2)}, \ldots, \psi_{H,i}^{(N_{H,i})}\right\} \subset M_{H,i}.$$

Remark 3.1. Note that each mortar basis function $\phi_{H,i}^{(k)}$ on interface $\Gamma_{i,j}$ corresponds to exactly two different Multiscale Flux Basis functions, one for Ω_i and one for Ω_j.

Remark 3.2. Whereas the original MMMFEM implementation requires each processor to perform the exact same number of subdomain solves during the interface iteration process, our implementation may have each processor perform a different number of subdomain solves in assembling its Multiscale Flux Basis. This is because there may be a varying number of degrees of freedom in each mortar $M_{H,i,j}$ and subdomains may share portions of their boundaries with Γ_D and Γ_N.

Remark 3.3. The Multiscale Flux Basis in this method consists of coarse interface fluxes, as opposed to fine scale subdomain data. The extra storage cost associated with saving this basis is equal to the square of the size of the mortar space on each subdomain, $N_{H,i}^2$. More specifically, each Multiscale Flux Basis function belongs to the mortar space $M_{H,i}$ of dimension $N_{H,i}$, and there are exactly $N_{H,i}$ basis functions to be computed. Therefore the storage cost for the Multiscale Flux Basis is significantly lower than the storage cost in the variational multiscale method and multiscale finite elements, where the basis functions are defined on the entire local fine grid.

3.2. Using the Multiscale Flux Basis in the interface iteration

To use the Multiscale Flux Basis to replace Steps 1–3 in the interface iteration, we need only observe that the flux operator $B_{H,i}$ is linear. Therefore,

$$\begin{aligned} B_{H,i}(\lambda_{H,i}) &= B_{H,i}\left(\sum_{k=1}^{N_{H,i}} \lambda_{H,i}^{(k)} \phi_{H,i}^{(k)}\right) = \sum_{k=1}^{N_{H,i}} \lambda_{H,i}^{(k)} B_{H,i}\left(\phi_{H,i}^{(k)}\right) \\ &= \sum_{k=1}^{N_{H,i}} \lambda_{H,i}^{(k)} \psi_{H,i}^{(k)}. \end{aligned} \tag{18}$$

In other words, to compute the resulting flux on subdomain Ω_i from Dirichlet data $\lambda_{H,i}$, we simply take a linear combination of the Multiscale Flux Basis functions $\psi_{H,i}^{(k)}$ using the same scalars which express $\lambda_{H,i}$ in terms of its mortar basis functions $\phi_{H,i}^{(k)}$. This demonstrates the equivalence of the original MMMFEM implementation to our new Multiscale Flux Basis implementation.

Remark 3.4. In the original MMMFEM implementation, fine scale pressure and velocity variables may also be updated iteratively in the interface iteration. In the new Multiscale Flux Basis implementation, this convention should be dropped, because storing arrays of these fine scale variables for each mortar degree of freedom would be an unnecessary burden on memory. Instead, we perform one additional Dirichlet subdomain solve after the CG iteration has converged in order to recover the fine scale pressure and velocity.

4. Numerical examples

The algorithm described in the previous section was implemented in the parallel flow simulator PARCEL, which is programmed in FORTRAN. The domain decomposition uses spatially conforming rectangular subdomains (in 2-D) or brick subdomains (in 3-D). Within each of these subdomains, the fine grid is comprised of the lowest order Raviart–Thomas–Nedelec mixed finite element spaces on rectangles (in 2-D) or bricks (in 3-D) [25,21], which are allowed to be spatially non-conforming across subdomain interfaces. On these subdomain interfaces, a coarse grid is comprised of continuous or discontinuous, linear or quadratic mortar spaces.

One of the goals of the numerical examples in this section is to compare the computational efficiency of the new Multiscale Flux Basis implementation to the original implementation. Since in the original implementation the number of interface iterations is directly related to the number of subdomain solves, we compare to both non-preconditioned and preconditioned methods. On the other hand, the Multiscale Flux Basis implementation shifts the workload from the number of interface iterations to the number of interface degrees of freedom per subdomain; hence we do not employ a preconditioner in this method. More precisely, the following three numerical methods are compared:

Method 1. Original MMMFEM implementation, no interface preconditioner.

Method 2. Original MMMFEM implementation, balancing preconditioner.

Method 3. New Multiscale Flux Basis implementation, no preconditioner.

Unless otherwise noted, the tolerance for the relative residual in the CG algorithm is taken to be 1e−06.

Remark 4.1. The balancing preconditioner used in the tests has been described in [13,22,7]. It involves solving Neumann subdomain problems and a course problem which provides global exchange of information across subdomains. This causes the condition number of the interface problem to grow more modestly versus non-preconditioned CG as the grids are refined or the number of subdomains increases. The cost for one preconditioned iteration is three subdomain solves and two coarse solves.

Four example problems are considered: a 2-D problem with smooth permeability, a 2-D problem with a rough permeability, a 3-D problem with smooth permeability, and a 2-D problem with adaptive mesh refinement. In the first three examples we solve each problem using a fixed fine grid several times. Each time we increase the number of subdomains, i.e. refine the coarse grid. This causes the interface problem to become larger and more ill-conditioned, hence increasing the number of CG iterations. Tables are provided which compare both the number of CG iterations and maximum number of subdomain solves required by the three methods.[1] In this way, the new Multiscale Flux Basis implementation can be directly compared to the original MMMFEM implementation. No error norms are reported in these tests, because all three methods produce the same solution within roundoff error.

For the first two examples we also provide tests comparing the accuracy and the cost of the MMMFEM solution to a fine scale solution.

The fourth example involves adaptive mesh refinement and illustrates the greater flexibility of the MMMFEM compared to existing multiscale methods. It also shows that the gain in efficiency from the new implementation is increased when grid adaptivity is employed.

Remark 4.2. It should be noted that under a fixed fine grid, as the number of subdomains is increased, the size of the local subdomain problems becomes smaller.

[1] Recall Remark 3.2.

Author's personal copy

4.1. Example 1: 2-D problem with a smooth solution

This example is a 2-D problem on the domain $\Omega = (0,1)^2$ with a fixed global fine grid of 120×120 elements. The solution is given by $p(x,y) = x^3y^4 + x^2 + \sin(xy)\cos(y)$, and the coefficient K is a smooth, full tensor defined by

$$K = \begin{pmatrix} (x+1)^2 + y^2 & \sin(xy) \\ \sin(xy) & (x+1)^2 \end{pmatrix}.$$

Boundaries $\{y = 0\}$ and $\{y = 1\}$ are Dirichlet type and boundaries $\{x = 0\}$ and $\{x = 1\}$ are Neumann type.

Table 1 shows results for Example 1 using continuous linear mortars with 3 elements per edge. Observe that the number of CG iterations increases with the number of subdomains, since the dimension of the interface problem grows. Recall that for all methods the dominant computational cost is measured by the number of subdomain solves. In Method 3 the number of subdomain solves does not depend on the number of interface iterations, only on the number of coarse scale mortar degrees of freedom per subdomain. As a result the number of subdomain solves does not change with increasing the number of subdomains (except for the 2×2 case where only two out of four edges of each subdomain have mortars). This is in contrast to the original implementation, Methods 1 and 2, where the number of subdomain solves is directly related to the number of CG interface iterations. Method 1 requires one subdomain solve per iteration plus three additional subdomain solves. Method 2 requires three subdomain solves per iteration plus ten additional subdomain solves. The balancing preconditioner used in Method 2 causes the number of CG iterations to grow more modestly with the number of subdomains, but this method is still more costly in terms of subdomain solves. Method 1 performs the best of all three methods until we reach the 4×4 subdomain case. After this point Method 3 becomes the most efficient in terms of subdomain solves. This table demonstrates that as the number of subdomains is increased, there is a point after which Method 3 performs best. We found this to be the case for most tests we ran.

Remark 4.3. Recall that the Balancing preconditioner involves two additional coarse grid solves per CG iteration. Thus even in cases where Method 2 required fewer subdomain solves, Method 3 was more efficient in terms of CPU time, as the time for the coarse solves was not negligible. We do not report CPU times in this paper, since they depend on the particular implementation of the coarse solve in the Balancing preconditioner.

Remark 4.4. There is also a cost in runtime associated with the interprocessor communication for each interface iteration. The Multiscale Flux Basis implementation does not reduce the number of interface iterations necessary for flux matching, hence it does not have an effect on the communication overhead. This cost can be reduced by using the Multiscale Flux Basis implementation in conjunction with an efficient preconditioner.

In Table 2 we report results for Example 1 with continuous quadratic mortars with 2 elements per edge. This slightly increases the required work for Method 1 and slightly decreases the work for Method 2. However, for Method 3 this change nearly doubles the amount of subdomain solves required due to the increase in mortar degrees of freedom per subdomain. This means that initially our method solves more subdomain problems than the other two, and the computational efficiency of Method 3 is not observed until the 5×5 case. This difference versus the previous table shows that the number of mortar degrees of freedom per subdomain is an important parameter which determines the relative computationally efficiency of the Multiscale Flux Basis implementation.

Table 1
Example 1 using continuous linear mortars with 3 elements per interface.

Subdomains	Method 1		Method 2		Method 3	
	CGIter	Solves	CGIter	Solves	CGIter	Solves
$2 \times 2 = 4$	14	17[a]	11	41	14	19
$3 \times 3 = 9$	29	32[a]	19	67	29	35
$4 \times 4 = 16$	42	45	24	82	42	35[a]
$5 \times 5 = 25$	54	57	26	88	54	35[a]
$6 \times 6 = 36$	65	68	27	91	64	35[a]
$7 \times 7 = 49$	75	78	26	88	77	35[a]
$8 \times 8 = 64$	86	89	26	88	86	35[a]

[a] Denotes fewest number of solves.

Table 2
Example 1 using continuous quadratic mortars with 2 elements per interface.

Subdomains	Method 1		Method 2		Method 3	
	CGIter	Solves	CGIter	Solves	CGIter	Solves
$2 \times 2 = 4$	16	19[a]	12	44	16	53
$3 \times 3 = 9$	35	38[a]	17	61	34	63
$4 \times 4 = 16$	51	54[a]	20	70	51	63
$5 \times 5 = 25$	65	68	21	73	65	63[a]
$6 \times 6 = 36$	78	81	22	76	78	63[a]
$7 \times 7 = 49$	91	94	21	73	91	63[a]
$8 \times 8 = 64$	103	107	21	73	103	63[a]

[a] Denotes fewest number of solves.

To illustrate the accuracy of the MMMFEM and the efficiency of the proposed new implementation, we compare the quality and cost of the multiscale solution to these of the fine scale solution. The latter is computed using the same domain decomposition algorithm with Method 3, but with fine scale Lagrange multipliers. In Table 3 we report the relative errors in pressure and velocity, and the cost of the interface iteration. This type of test is comparable to a standard mixed finite element algorithm without domain decomposition. Indeed, the recorded error norms remain nearly constant as the number of subdomains is increased.

In comparison, Table 4 shows results for the MMMFEM using Method 3 with linear mortars and a single element per interface. This subdomain configuration is very much akin to the variational multiscale methods and multiscale finite element methods mentioned in the introduction. We note that the MMMFEM requires significantly smaller number of subdomain solves, while at the same time resolves the flow very well, as seen in Fig. 2 where a

Table 3
Relative errors and computational cost for the fine scale solution in Example 1.

Subdomains	pres-L2-err	vel-L2-err	CGIter	Solves
$2 \times 2 = 4$	7.1657E−05	7.1848E−05	58	123
$3 \times 3 = 9$	7.1955E−05	7.9269E−05	72	163
$4 \times 4 = 16$	7.1968E−05	8.7311E−05	85	123
$5 \times 5 = 25$	7.2211E−05	9.5265E−05	96	99
$6 \times 6 = 36$	7.2260E−05	1.0281E−04	107	83

Table 4
Relative error and computational cost for the multiscale solution using Method 3 with a single linear mortar per interface in Example 1.

Subdomains	pres-L2-err	vel-L2-err	CGIter	Solves
$2 \times 2 = 4$	1.2966E−02	4.4386E−02	8	11
$3 \times 3 = 9$	7.1036E−03	3.6534E−02	22	19
$4 \times 4 = 16$	4.2496E−03	3.0038E−02	33	19
$5 \times 5 = 25$	2.7673E−03	2.5191E−02	42	19
$6 \times 6 = 36$	1.9159E−03	2.1527E−02	51	19

Author's personal copy

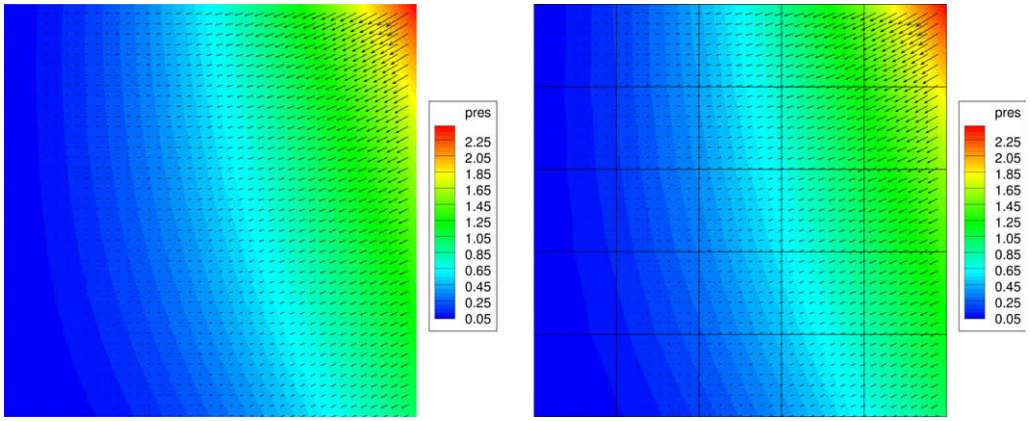


Fig. 2. Computed pressure (color) and velocity (arrows) in Example 1: fine scale solution (left) and multiscale solution with a single linear mortar per interface (right).

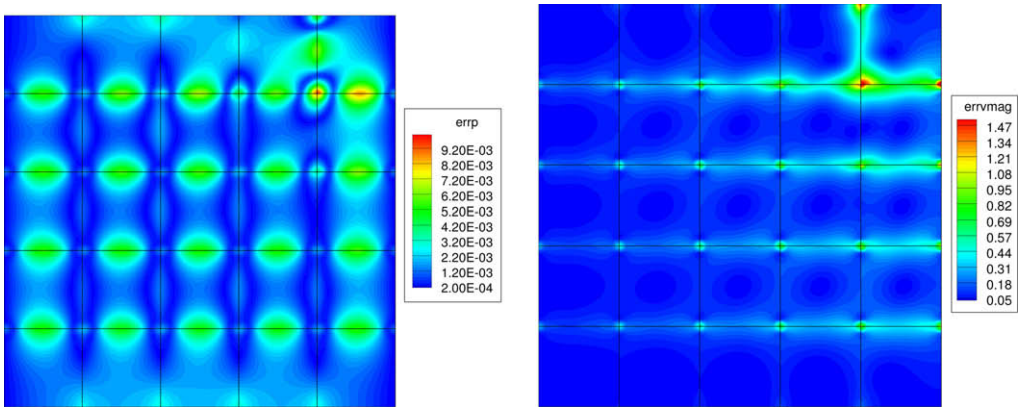


Fig. 3. Pressure error (left) and magnitude of velocity error (right) for the multiscale solution with a single linear mortar per interface in Example 1.

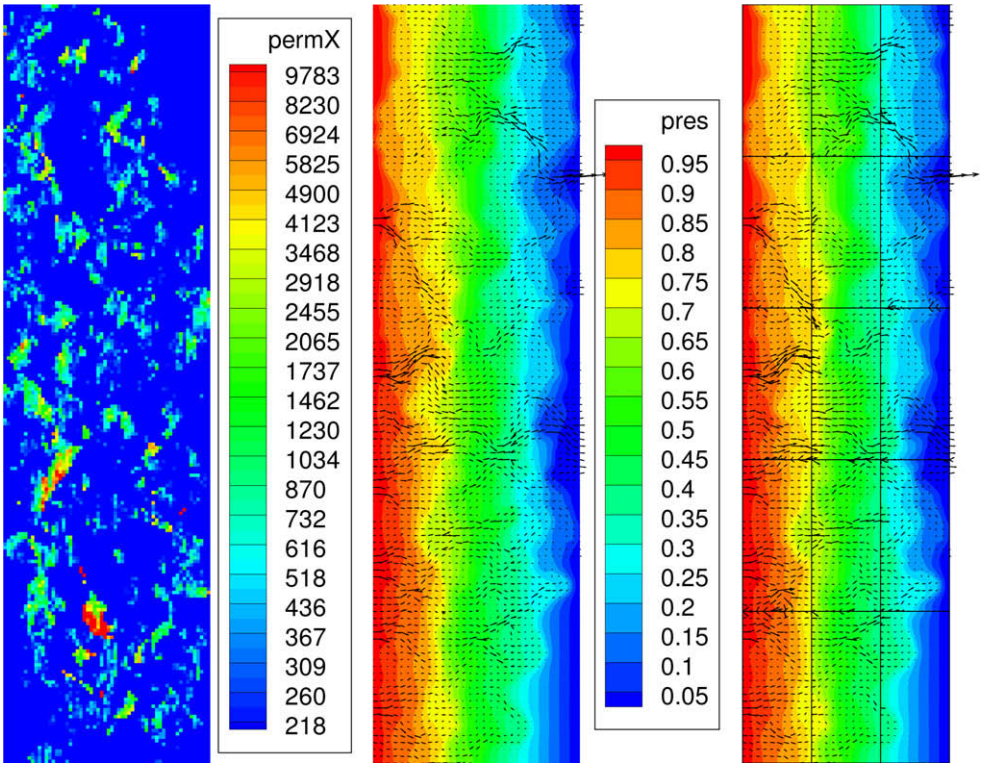


Fig. 4. Example 2: permeability field (left), fine scale solution (middle), and multiscale solution with 3×5 subdomains and a single linear mortar per interface (right).

Author's personal copy

Table 5
Example 2 using continuous linear mortars with 2 elements per interface.

Subdomains	Method 1		Method 2		Method 3	
	CGIter	Solves	CGIter	Solves	CGIter	Solves
$2 \times 2 = 4$	13	15	8	31	13	14[a]
$3 \times 2 = 6$	19	21	15	53	19	20[a]
$2 \times 4 = 8$	25	27	18	62	23	20[a]
$2 \times 5 = 10$	37	39	29	95	35	20[a]
$3 \times 4 = 12$	37	39	28	93	36	26[a]
$3 \times 5 = 15$	51	53	37	120	51	26[a]

[a] Denotes fewest number of solves.

Table 6
Example 2 using continuous quadratic mortars with 2 elements per interface.

Subdomains	Method 1		Method 2		Method 3	
	CGIter	Solves	CGIter	Solves	CGIter	Solves
$2 \times 2 = 4$	17	19[a]	15	52	16	32
$3 \times 2 = 6$	30	32[a]	23	77	31	47
$2 \times 4 = 8$	39	41[a]	25	83	38	47
$2 \times 5 = 10$	56	58	39	125	56	47[a]
$3 \times 4 = 12$	53	55	33	108	52	62[a]
$3 \times 5 = 15$	92	94	46	147	92	62[a]

[a] Denotes fewest number of solves.

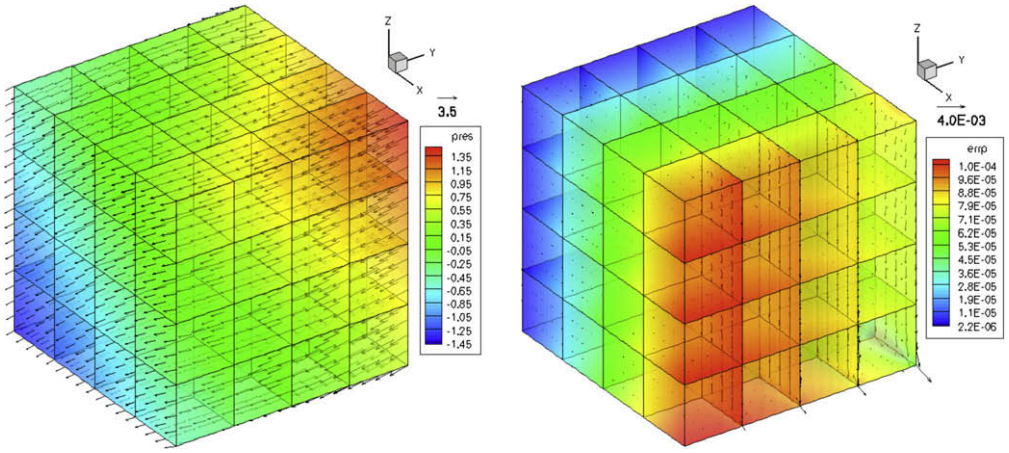


Fig. 5. Example 3: computed multiscale solution (left) and its error (right) on $4 \times 4 \times 4$ subdomains with a single linear mortar per interface.

comparison of the plots of the computed fine scale and multiscale solutions with 5×5 subdomains is shown. The relative error norms reported in Table 4 indicate that the error is larger for the multiscale solution, but does decrease as the number of subdomain is increased. Fig. 3 shows that the locations with greatest error in the multiscale solution are along the subdomain interfaces.

4.2. Example 2: 2-D problem with rough heterogeneous permeability

This problem uses a 2-D heterogeneous permeability field, obtained from the Society of Petroleum Engineers (SPE) Comparative Solution Project.[2] The domain is $\Omega = (0,60) \times (0,220)$ with a fixed global fine grid of 60×220 elements. Pressure values of one and zero are specified on the left and right boundaries, respectively. No flow is specified on the top and bottom boundaries. A plot of the permeability field is shown on the left in Fig. 4.

Table 5 shows the results for Example 2 using continuous linear mortars with 2 elements per edge. Method 3 requires at most 26 solves per subdomain and is computationally more efficient than Methods 1 and 2 for all subdomain configurations. As the number of subdomains is increased, the improvement of Method 3 over Methods 1 and 2 becomes greater.

A comparison between the fine scale solution and the multiscale solution with 3×5 subdomains is presented in Fig. 4. We observe a very good match between the two solutions. We note that the number of subdomain solves required by Method 3 for the multiscale solution, 26, is significantly less than Methods 1, 2, 3 used for computing the fine scale solution, which require 388, 84, and 130 subdomain solves, respectively.

[2] For more information, see http://www.spe.org/csp.

Table 6 shows the results for Example 2 using continuous quadratic mortars with 2 elements per interface. Compared to the previous table, the increased number of mortar degrees of freedom per interface leads to more subdomain solves for Method 3, the maximum number being 62. Nevertheless, Method 3 is still more computationally efficient than Methods 1 and 2 for 10 and more subdomains.

4.3. Example 3: 3-D problem with a smooth solution

This example is a 3-D problem on the domain $\Omega = (0,1)^3$ with a fixed global fine grid of $48 \times 48 \times 48$ elements. The solution is given by $p(x,y,z) = x + y + z - 1.5$, and the coefficient K is a smooth full tensor defined by

$$K = \begin{pmatrix} x^2+y^2+1 & 0 & 0 \\ 0 & z^2+1 & \sin(xy) \\ 0 & \sin(xy) & x^2y^2+1 \end{pmatrix}.$$

Boundaries $\{y = 0\}$ and $\{y = 1\}$ are Dirichlet type and the rest of the boundary is Neumann type.

Fig. 5 shows the computed multiscale solution and its error for Example 3 with $4 \times 4 \times 4$ subdomains and a single linear mortar per interface. Table 7 shows the computational cost for Methods 1, 2, 3 with various coarse grids. Method 3 requires at most 27 solves per subdomain and outperforms Methods 1 and 2 for all subdomain configurations.

Table 8 shows the results for Example 3 using quadratic mortars with one element per interface with the usual relative residual CG tolerance of 1e−06. Method 3 requires at most 57 solves per subdomain. It is the fastest method on coarser domain decompositions, but Method 2 outperforms it slightly on 27 or more subdomains.

Author's personal copy

Table 7
Example 3 using linear mortars with one element per interface.

Subdomains	Method 1		Method 2		Method 3	
	CGIter	Solves	CGIter	Solves	CGIter	Solves
$2 \times 2 \times 2 = 8$	28	31	11	42	28	15[a]
$2 \times 2 \times 3 = 12$	33	36	12	46	33	19[a]
$2 \times 3 \times 3 = 18$	37	40	13	50	37	23[a]
$3 \times 3 \times 3 = 27$	46	49	13	51	46	27[a]
$3 \times 3 \times 4 = 36$	50	53	13	51	50	27[a]
$3 \times 4 \times 4 = 48$	55	58	13	51	55	27[a]
$4 \times 4 \times 4 = 64$	60	63	13	51	60	27[a]

[a] Denotes fewest number of solves.

Table 8
Example 3 using quadratic mortars with one element per interface. Relative residual CG tolerance = 1e−06.

Subdomains	Method 1		Method 2		Method 3	
	CGIter	Solves	CGIter	Solves	CGIter	Solves
$2 \times 2 \times 2 = 8$	36	39	13	48	36	30[a]
$2 \times 2 \times 3 = 12$	41	44	13	49	41	39[a]
$2 \times 3 \times 3 = 18$	47	50	14	53	47	48[a]
$3 \times 3 \times 3 = 27$	56	59	14	54[a]	56	57
$3 \times 3 \times 4 = 36$	60	63	14	54[a]	60	57
$3 \times 4 \times 4 = 48$	64	67	14	54[a]	64	57
$4 \times 4 \times 4 = 64$	69	72	14	54[a]	69	57

[a] Denotes fewest number of solves.

Table 9
Example 3 using quadratic mortars with one element per interface. Relative residual CG tolerance = 1e−09.

Subdomains	Method 1		Method 2		Method 3	
	CGIter	Solves	CGIter	Solves	CGIter	Solves
$2 \times 2 \times 2 = 8$	48	51	19	66	48	30[a]
$2 \times 2 \times 3 = 12$	56	59	19	67	56	39[a]
$2 \times 3 \times 3 = 18$	62	65	21	74	62	48[a]
$3 \times 3 \times 3 = 27$	74	77	20	72	74	57[a]
$3 \times 3 \times 4 = 36$	79	82	21	75	79	57[a]
$3 \times 4 \times 4 = 48$	84	87	21	75	85	57[a]
$4 \times 4 \times 4 = 64$	92	95	21	75	92	57[a]

[a] Denotes fewest number of solves.

When a tighter tolerance is imposed on the CG on the interface, all three methods perform more CG iterations. Under Methods 1 and 2, this also requires performing more subdomain solves. For Method 3, however, the maximum number of solves per subdomain is unaffected by this change in tolerance. This is illustrated in Table 9, which shows the results for relative residual CG tolerance of 1e−09. In this case Method 3 is the most computationally efficient for all subdomain configurations.

4.4. Example 4: mesh adaptivity

The final example illustrates an increased computational benefit from the MMMFEM when adaptive mesh refinement is utilized. By using the Multiscale Flux Basis implementation on each refinement level, the overall computational savings are compounded. In this example, residual based *a posteriori* error indicators are used to refine only those subdomains where the error is highest. Mortars that touch refined subdomains are also refined in order to maintain accuracy. This approach has been shown to be both efficient and reliable, see [27,7] for details.

The permeability K is a single realization of a stochastic permeability field on the domain $(0,1)^2$. A Karhunen–Loève (KL) expansion for the log permeability $Y = \ln(K)$ (a scalar quantity) is computed from the specified covariance function

$$C_Y(\mathbf{x}, \bar{\mathbf{x}}) = \sigma_Y^2 \exp\left[\frac{-|x_1 - \bar{x}_1|}{\eta_1} - \frac{|x_2 - \bar{x}_2|}{\eta_2}\right].$$

The parameters used for this test are correlation lengths $\eta_1 = 0.25, \eta_2 = 0.125$, and variance $\sigma_Y = 2.1$. The series was truncated after 400 terms.

We specifically chose to generate the permeability in this example as a realization of a KL expansion, so that no upscaling or homogenization would be needed. On each level of mesh refinement, we are able to analytically evaluate a very heterogeneous permeability on an arbitrarily fine grid. For the procedure for computing the eigenfunctions and eigenvalues of this series, the interested reader can consult Appendix A in [29].

This test was performed on $5 \times 5 = 25$ subdomains, initially starting with 2×2 subdomain grids and continuous linear mortars with one element per edge. The permeability field and its corresponding solution on the fourth level of mesh refinement are shown in Fig. 6. Note that this adaptive procedure leads to different scales being resolved on different subdomains, providing a truly multiscale approximation. One can see the subdomains now have 4×4, 8×8, and 16×16 grids.

Using Method 1, each subdomain performed 283 subdomain solves, roughly one for each CG iteration on each of the 4 grid levels. Using Method 3, the number of subdomain solves after 4 levels of mesh refinement is shown in the figure on top of the permeability plot. The maximum number of subdomain solves is 160 and the minimum number is 56.

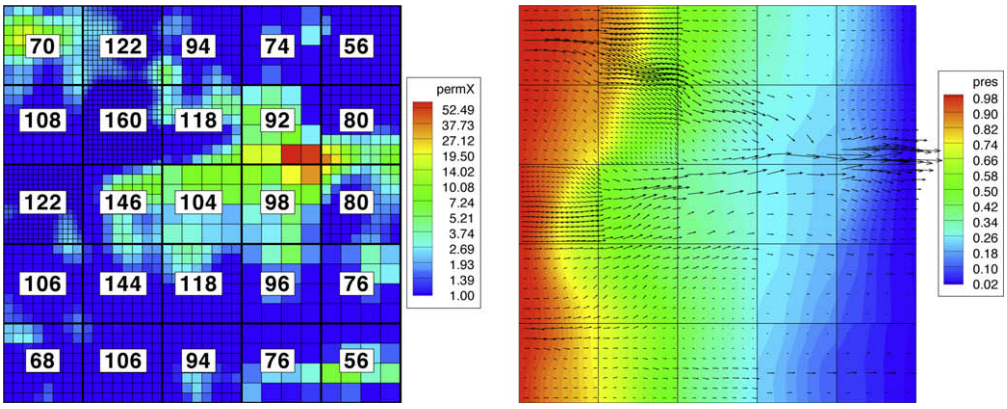


Fig. 6. Permeability field for Example 4 on mesh refinement level 4 (left) and its corresponding solution (right). Numbers indicate the total number of subdomain solves using the Multiscale Flux Basis implementation.

Author's personal copy

We can draw two conclusions from this example. First, since the computational savings of the Multiscale Flux Basis implementation happen on each level of adaptive mesh refinement, the overall savings after all levels are complete is amplified by the number of refinement levels. Second, the workload for each processor may become increasingly unbalanced due to a large variation in the number of mortar degrees of freedom per subdomain. Nevertheless, even if the algorithm is only as fast as its slowest processor, the Multiscale Flux Basis implementation is still faster than the original implementation. One can take full advantage of the computational efficiency of the new method in adaptive mesh refinement setting by implementing load balancing.

5. Conclusions

In this paper we present a new implementation of the MMMFEM, which makes it comparable in computational cost to existing multiscale methods. The MMMFEM provides extra flexibility in the ability to vary locally and adaptively both the coarse scale and the fine scale grids. Moreover, the fine scale grids can be completely non-matching across coarse interfaces. The proposed implementation is based on precomputing a Multiscale Flux Basis and using it to compute solutions to subdomain solves during the global coarse scale iteration.

The numerical examples demonstrate that the new implementation is more computationally efficient than the original implementation in many cases. The number of subdomain solves required for the construction of the Multiscale Flux Basis depends only on the number of mortar degrees of freedom per subdomain and not on the size of the global problem. Therefore the new implementation outperforms the original one for large problems. Moreover, if the Multiscale Flux Basis implementation is used repeatedly, as in the case of adaptive mesh refinement, then the computational savings are amplified. Even greater computational gain is observed when this approach is combined with stochastic collocation for uncertainty quantification, which requires a large number of deterministic simulations. This extension will be discussed in a forthcoming paper.

Acknowledgments

This work was partially supported by the NSF Grants DMS 0620402 and DMS 0813901 and the DOE Grant DE-FG02-04ER25618. We would also like to thank the Institute for Computational Engineering and Sciences at the University of Texas at Austin for the use of their computing resources.

References

[1] J.E. Aarnes, Y. Efendiev, L. Jiang, Mixed multiscale finite element methods using limited global information, Multiscale Model. Simul. 7 (2) (2008) 655–676.
[2] Jørg 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 (3) (2004) 421–439.
[3] Jørg Aarnes, Stein Krogstad, Knut-Andreas Lie, A hierarchical multiscale method for two-phase flow based upon mixed finite elements and nonuniform coarse grids. Multiscale Model. Simul. 5 (2) (2007) 337–363.
[4] T. Arbogast, Analysis of a two-scale, locally conservative subgrid upscaling for elliptic problems, SIAM J. Numer. Anal. 42 (2004) 576–598.
[5] T. Arbogast, K.J. Boyd, Subgrid upscaling and mixed multiscale finite elements, SIAM J. Numer. Anal. 44 (3) (2007) 1150–1171.
[6] T. Arbogast, L.C. Cowsar, M.F. Wheeler, I. Yotov, Mixed finite element methods on nonmatching multiblock grids, SIAM J. Numer. Anal. 37 (2000) 1295–1315.
[7] T. Arbogast, G. Pencheva, M.F. Wheeler, I. Yotov, A multiscale mortar mixed finite element method, Multiscale Model. Simul. 6 (1) (2007) 319.
[8] F. Brezzi, J. Douglas Jr., R. Duràn, M. Fortin, Mixed finite elements for second order elliptic problems in three variables, Numer. Math. 51 (1987) 237–250.
[9] F. Brezzi, J. Douglas Jr., M. Fortin, L.D. Marini, Efficient rectangular mixed finite elements in two and three space variables, RAIRO Modèl. Math. Anal. Numèr. 21 (1987) 581–604.
[10] F. Brezzi, J. Douglas Jr., L.D. Marini, Two families of mixed elements for second order elliptic problems, Numer. Math. 88 (1985) 217–235.
[11] Z. Chen, T.Y. Hou, A mixed multiscale finite element method for elliptic problems with oscillating coefficients, Math. Comp. 72 (2003) 541–576.
[12] Zhangxin Chen, J. Douglas Jr., Prismatic mixed finite elements for second order elliptic problems, Calcolo 26 (1989) 35–148.
[13] L.C. Cowsar, J. Mandel, M.F. Wheeler, Balancing domain decomposition for mixed finite elements, Math. Comp. 64 (1995) 989–1015.
[14] Yalchin R. Efendiev, Thomas Y. Hou, Xiao-Hui Wu, Convergence of a nonconforming multiscale finite element method, SIAM J. Numer. Anal. 37 (3) (2000) 888–910 (electronic).
[15] R. Glowinski, M.F. Wheeler, Domain decomposition and mixed finite element methods for elliptic problems, in: First International Symposium on Domain Decomposition Methods for Partial Differential Equations, Philadelphia, PA, 1988.
[16] T.Y. Hou, X.H. Wu, A multiscale finite element method for elliptic problems in composite materials and porous media, J. Comput. Phys. 134 (1997) 169–189.
[17] T.Y. Hou, X.-H. Wu, Z. Cai, Convergence of a multiscale finite element method for elliptic problems with rapidly oscillating coefficients, Math. Comp. 68 (1999) 913–943.
[18] 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) 387–401.
[19] T.J.R. Hughes, G.R. Feijóo, L. Mazzei, J.-B. Quincy, The variational multiscale method – a paradigm for computational mechanics, Comput. Methods Appl. Mech. Engrg. 166 (1998) 3–24.
[20] P. Jenny, S.H. Lee, H.A. Tchelepi, Multi-scale finite-volume method for elliptic problems in subsurface flow simulation, J. Comp. Phys. 187 (2003) 47–67.
[21] J.C. Nedelec, Mixed finite elements in R^3, Numer. Math. 35 (1980) 315–341.
[22] G. Pencheva, I. Yotov, Balancing domain decomposition for mortar mixed finite element methods, Numer. Linear Algebra Appl. 10 (2003) 159–180.
[23] Małgorzata Peszyńska, Mary F. Wheeler, Ivan Yotov, Mortar upscaling for multiphase flow in porous media, Comput. Geosci. 6 (1) (2002) 73–100.
[24] A. Quarteroni, A. Valli, Numerical Approximation of Partial Differential Equations, Springer, 1994.
[25] R.A. Raviart, J.M. Thomas, A mixed finite element method for 2nd order elliptic problems, in: Mathematical Aspects of the Finite Element Method, Lecture Notes in Mathematics, vol. 606, Springer-Verlag, New York, 1977, pp. 292–315.
[26] M.F. Wheeler, I. Yotov, Physical and computational domain decompositions for modeling subsurface flows, in: Jan Mandel et al. (Eds.), Tenth International Conference on Domain Decomposition Methods, Contemporary Mathematics, vol. 218, American Mathematical Society, 1998, pp. 217–228.
[27] Mary F. Wheeler, Ivan Yotov, A posteriori error estimates for the mortar mixed finite element method, SIAM J. Numer. Anal. 43 (3) (2005) 1021–1042.
[28] I. Yotov, Mixed Finite Element Methods for Flow in Porous Media, Ph.D. Thesis, Rice University, 1996.
[29] D. Zhang, Z. Lu, An efficient, high-order perturbation approach for flow in random porous media via Karhunen–Loève and polynomial expansions, J. Comput. Phys. 194 (2) (2004) 773–794.