AN OpenMP IMPLEMENTATION OF THE MuMM SOLVER FOR POROUS MEDIA FLOW PROBLEMS

UMA IMPLEMENTAÇÃO OpenMP DO RESOLUTOR MuMM PARA PROBLEMAS DE ESCOAMENTO EM MEIO POROSOS

Abstract: Multiscale methods are usually developed for solving second-order elliptic problems in which coefficients are of multiscale heterogeneous nature. The Multiscale Mixed Method (MuMM) was introduced aiming at the efficient and accurate approximation of large flow problems in highly heterogeneous porous media. In the MuMM numerical solver, first mixed multiscale basis functions are constructed, and next global domain decomposition iterations are performed to compute the discrete solution of the problems. However, this iterative procedure is a time-consuming step. In this paper, the authors improve the MuMM solver through the implementation of parallel computations in the step concerning the global iterative procedure. The parallel version of the solver employs the application programming interface Open Multi-Processing (OpenMP). The implementation with the OpenMP reduces significantly the computational effort to perform the domain decomposition iterations, as indicated by the numerical results.

Keywords: Parallel computing. Multiscale method. Porous media.

1Mestre em Engenharia Mecânica, Universidade Federal Fluminense, sergio_felipe@id.uff.br.
2Graduada em Engenharia Mecânica, Universidade Federal Fluminense, hannaarimateia@id.uff.br.
3Doutor em Engenharia Nuclear, Universidade Federal Fluminense, afrancisco@id.uff.br.
4Doutor em Engenharia Metalúrgica, Universidade Federal Fluminense, wesleyassis@id.uff.br.

Revista Mundi, Engenharia e Gestão, Paranaguá, PR, v. 5, n. 6, p. 284-01, 284-14, 2020 DOI: 10.21575/25254782rmetag2020vol5n61333
Resumo: Métodos multiescala podem ser usualmente desenvolvidos para resolver problemas elípticos de segunda ordem, nos quais coeficientes são de natureza heterogênea multiescala. O Método Misto Multiescala (MuMM) foi introduzido visando a aproximação eficiente e acurada de extensos problemas de escoamento em meios porosos. No resolutor numérico MuMM, primeiramente funções-base mista multiescala são construídas, e então interações de decomposição de domínio global são executadas para computar a solução discreta dos problemas. Entretanto, esse procedimento iterativo é um passo que consome tempo considerável. Neste trabalho, os autores aperfeiçoam o resolutor MuMM por meio da implementação de computação paralela no passo que diz respeito ao procedimento iterativo global. A versão paralela do resolutor emprega a interface de programação aplicativa Open Multi-Processing (OpenMP). A implementação com o OpenMP proporciona um esforço computacional reduzido para executar o procedimento iterativo global, conforme indicado pelos resultados numéricos.

1 INTRODUCTION

The development of multiscale numerical methods for second-order elliptic differential equations arising in porous media flow problems has attracted the attention of several research groups. The reader is referred to (CHU et al., 2008; EFENDIEV; HOU, 2009; KIPPE; AARNES; LIE, 2008; WHEELER; WILDEY; XUE, 2010) in order to see of the related procedures to address these problems.

In this paper, the authors focus on the Multiscale Mixed Method (MuMM), a numerical solver that uses a non-overlapping domain decomposition iterative procedure in which the spatial discretization of local problems uses the hybridized mixed finite elements at fine scale and the Robin boundary condition at coarse scale (FRANCISCO et al., 2014). Mixed multiscale basis functions are used to compute the discrete solutions in local problems. The analogous use of the concept of multiscale flux basis functions can be found in (GANIS; YOTOV, 2009). The MuMM solver provides fast and accurate approximation for second-order elliptic equations. The multiscale procedure can take the advantage of heterogeneous processing units, which are relatively inexpensive and have great computational power.

An enhancement of the MuMM solver is done by implementing parallel computing in the algorithm. The parallel algorithm of this solver takes the advantage of the application programming interface Open Multi-Processing (OpenMP). The implementation with OpenMP reduces significantly the computational effort to generate a good quality numerical approximation of the solution at very fine scale (CHAPMAN; JOST; PAS, 2008).

This paper is organized as follows. Section 2 describes the multiscale procedure for the porous media flow problem, and shortly gives the variational formulation involved in the construction of mixed multiscale basis functions and iterative procedure. In Section 3 the parallel implementation by using the OpenMP of the MuMM solver is described. In Section 4 the numerical results from the implementation are discussed. Finally, in Section 5, the paper present the concluding remarks.

2 THE GLOBAL PROBLEM

The single-phase, incompressible flow in porous media is modeled by the pressure-velocity system of equations on a bounded domain $\Omega$ (CHEN; HUAN; MA, 2006) that is expressed by

$$\nabla \cdot u = q, \quad \text{and} \quad u = -K(x)\nabla p, \quad (1)$$
where $K$ is the permeability divided by the viscosity, $u$ is the Darcy velocity, $p$ is the pressure, and $q$ is the source term. Dirichlet and Neumann boundary conditions on $\partial\Omega$ are respectively expressed as

$$p = p_b \text{ on } \Gamma_D \text{ and } u \cdot \nu = u_b \text{ on } \Gamma_N,$$

(2)

where $\partial\Omega = \bar{\Gamma}_D \cup \bar{\Gamma}_N$, $\Gamma_D \cap \Gamma_N = \phi$, and $\nu$ is the outward unit normal vector to $\partial\Omega$.

3 THE MuMM SOLVER

A rectangular domain $\Omega$ can be decomposed into uniform squares according to: $i$) the fine mesh of length scale $h$; $ii$) the intermediate mesh of length scale $H$; and $iii$) the coarse mesh of length scale $H$. Distinct square meshes are illustrated in Fig. 1.

Over the decomposition, fine and coarse rectangular elements $\Omega^\gamma_j$, $j = 1, ..., M^\gamma$, $\gamma = f, c$, are constructed such that $\bar{\Omega} = \bigcup_{j=1}^{M^f} \bar{\Omega}_j$, $\Omega^\gamma_j \cap \Omega^\gamma_k = \phi$, $j \neq k$. The interface between an element and another adjacent is denoted by $\Gamma^\gamma_{jk} = \partial \Omega^\gamma_j \cap \partial \Omega^\gamma_k$.

3.1 Global Coarse-Mesh Decomposition

First, a global coarse-mesh decomposition is done. Thus, local problems are posed in coarse elements $\Omega^c_m$, where the pressure and velocity spaces are defined respectively as $W = L^2(\Omega^c_m)$ and $V = H(div; \Omega^c_m)$. The mixed finite element formulation for a local problem is to find $\{u_m, p_m\} \in W \times V$, such that (JR. et al., 1993)

$$(\nabla \cdot u_m, w)_{\Omega^c_m} = (q, w)_{\Omega^c_m}, \ w \in W,$$

(3)
\[(K^{-1}(x)u_m, \mu)_{\Omega^c_m} - (p_m, \nabla \cdot \mu)_{\Omega^c_m} + (a u_m \cdot \nu + g, \mu \cdot \nu)_{\partial \Omega^c_m} = 0, \quad \mu \in V \] (4)

with the imposition of the generic Robin boundary condition

\[-a u_m \cdot \nu + p_m = g \text{ on } \partial \Omega^c_m, \] (5)

where \(a\) and \(g\) are given functions.

### 3.2 Mixed Multiscale Basis Functions

Let \(\Gamma^c_{mn} = \partial \Omega^c_m \cap \partial \Omega^c_n, \quad n = r, l, b, t\), denote the interfaces between an element and the other four adjacents. For efficiency purposes, the intermediate length scale \(\bar{H} (h \leq \bar{H} \leq H)\) is introduced, and each interface \(\Gamma^c_{mn}\) is decomposed into line segments \(I_{mn}^i, i = 1, \ldots, \bar{r}\), such that \(\bar{r} = H/\bar{H}\).

Next, divergence free local problems are posed in the coarse elements \(\Omega^c_m\). The mixed finite element formulation for a new local problem is to find \(\{\psi^i_{mn}, \phi^i_{mn}\} \in W \times V\), such that

\[(\nabla \cdot \psi^i_{mn}, w)_{\Omega^c_m} = 0, \quad w \in W, \] (6)

\[(K^{-1}(x)\psi^i_{mn}, \mu)_{\Omega^c_m} - (\phi^i_{mn}, \nabla \cdot \mu)_{\Omega^c_m} + (a_{mn}^i \psi^i_{mn} \cdot \nu + g_{mn}^i, \mu \cdot \nu)_{\partial \Omega^c_m} = 0, \quad \mu \in V \] (7)

with the imposition of the Robin boundary condition

\[-a_{mn}^i \psi^i_{mn} \cdot \nu + \phi^i_{mn} = g_{mn}^i \text{ on } \partial \Omega^c_m, \] (8)

where

\[g_{mn}^i = \begin{cases} 1, & \text{on } I_{mn}^i \\ 0, & \text{otherwise.} \end{cases} \] (9)

Each solution \(\{\psi^i_{mn}, \phi^i_{mn}\}\) determined at fine elements \(\Omega^f_{mn}\) defines the mixed multiscale basis functions tied to one \(I_{mn}^i\).
3.3 Global Coarse-Mesh Iteration

For solving numerically the Eqs. (3) and (4), the \textit{MuMM} solver proceeds a global iteration in which the current solution in local problems $\Omega^c_m$ is computed by the following linear combination (FRANCISCO et al., 2014):

\begin{equation}
\mathbf{u}_m = \sum_n \sum_i \bar{g}^i_{mn} \psi^i_{mn},
\end{equation}

\begin{equation}
\mathbf{p}_m = \sum_n \sum_i \bar{g}^i_{mn} \phi^i_{mn},
\end{equation}

where $\bar{g}^i_{mn}$ are coefficients defined by averaging the Robin boundary condition on each $I^i_{mn}$, at previous iteration, that is

\begin{equation}
\bar{g}^i_{mn} = \frac{1}{H} \int_{I^{imn}_{mn}} (\bar{a}^i_{mn} \mathbf{u}_n \cdot \nu + p_n) dl,
\end{equation}

where $\bar{a}^i_{mn}$ is a positive constant on each $I^{i}_{mn}$.

3.4 Downscaling Solution

After the convergence of the global iteration procedure is achieved, the flux conservation at coarse mesh are hold. To get the flux conservation at fine mesh, local problems must be solved with the imposition of Neumann boundary conditions, where the boundary values are specified by averaging the velocities on each $I^{i}_{mn}$. With this downscaling procedure, the discrete solution at fine mesh is found for the global problem.

4 OpenMP IMPLEMENTATION

In summary, the MuMM solver was developed by using a non-overlapping iterative decomposition procedure with the Robin boundary condition at coarse mesh. Mixed multiscale basis functions was constructed to represent the solutions in local problems at fine mesh, using hybridized mixed finite elements for the spatial discretization. Figure 2 displays the flowchart of the \textit{MuMM} solver through stages and corresponding code functions.

In order to identify loops in which CPU time is longer, a computational analysis is carried out by using the Performance Profiler of the Visual Studio. The effect of the heterogeneity in the permeability field on the CPU time consumption of the code functions is also checked and
Figure 2: The flowchart of the MuMM solver.

Source: Author (2020).

showed in Table 1. The loops in the GLOBAL_SOLVER are found with the longer CPU time. It is observed that the CPU time consumption of the ITERATION_SOLVER is meaningful when the permeability field is heterogeneous.

Table 1: CPU time consumed by MuMM solver

<table>
<thead>
<tr>
<th>Function</th>
<th>CPU time (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Homogeneous</td>
</tr>
<tr>
<td>GLOBAL_SOLVER</td>
<td>90.0</td>
</tr>
<tr>
<td>BASIS_SOLVER</td>
<td>&lt; 1.0</td>
</tr>
<tr>
<td>ITERATION_SOLVER</td>
<td>&lt; 1.0</td>
</tr>
<tr>
<td>NEUMANN_SOLVER</td>
<td>&lt; 0.1</td>
</tr>
<tr>
<td>Others</td>
<td>&lt; 8.0</td>
</tr>
</tbody>
</table>
A parallel algorithm is developed and implemented in the computational code by using OpenMP directives. To avoid performing test cases with several construct and clause combinations, the parallel for construct and specific clauses are considered the best choices for the GLOBAL_SOLVER and ITERATION_SOLVER. The OpenMP parallel implementation in the code functions are respectively described in Algorithms 1 and 2.

Algorithm 1: GLOBAL_SOLVER.

1. Set the tolerance TOL.
2. Set the null initial guess at center of all elements.
4. while NORM > TOL do
5.     Compute the solution at center of down elements;
6.     Compute the solution at center of up elements;
7.     Compute the solution at center of interior elements; /* #pragma omp parallel for num_threads(...) schedule(dynamic,8) default(shared) private(...) */
8.     Calculate NORM for checking the convergence.
9.     Save the solution at center of all elements; /* #pragma omp parallel for num_threads(...) schedule(dynamic,8) default(shared) private(...) */
10.    firstprivate(...)
11. Compute the solution at center and edge of all elements.
5 RESULTS AND DISCUSSIONS

In order to implement the OpenMP in the MuMM solver, simulations for the single-phase, incompressible flow in porous medium in SLAB geometry are performed. The SLAB
The problem is illustrated in Fig. 3. The physical domain has $12800 \times 12800$ $m^2$. The left and right boundary conditions are of Dirichlet type with $p_b = 1.0$ $Pa$ (Injection) or $p_b = 0$ $Pa$ (Production), respectively. The top and bottom boundary conditions are of Neumann type with $u_b = 0$ (No flow). There is no source term ($q = 0$), and the average permeability is $2.0 \times 10^{-11}$ $m^2$. For all the simulations, $\bar{H} = H$.

![Figure 3: The SLAB problem.](image)

Source: Author (2020).

The computational apparatus used for all the simulations is a Windows 10 Pro-personal computer, with 24 GB available RAM, and Intel Xeon CPU E5520. This processor has 2 physical processors of 2.27 GHz, with 4 cores each one. Therefore, 8 threads are to be used in the parallel simulation. The Intel C++ Compiler 19.0 is used inside Microsoft Visual Studio Community 2019 IDE. The compiler flags used for all the simulations is Generate Parallel Code (/Qopenmp) and Maximum Optimization (/O2). Other default optimizations and flags were preserved.

5.1 Sequential Algorithm

In order to get the best balance between fine and coarse meshes, simulations of the flow problem in homogeneous porous media are performed. The best multiscale mesh arrangement is the one that presents smaller global error when compared to a reference solution. For benchmark purposes, the problem is solved by a mixed finite element method ($MFEM$), which is able to provide accurate velocity fields even for highly heterogeneous porous media (JR. et al., 1993).

This numerical experiment is performed using a sequential algorithm of the $MuMM$ solver, considering the fine mesh of $512 \times 128$ elements and the coarse mesh of $16 \times 4, 32 \times 8, 64 \times 16, 128 \times 32$ and $256 \times 64$ elements. For the $MFEM$ solver, it is considered the fine mesh of $512 \times 128$ elements. The CPU time consumption is obtained by using the $omp_get_wtime$ function, and it is measured during the entire code running.
Figure 4 shows the results of global error for the velocity and pressure fields. As the coarse mesh is refined, the global error for both the fields is reduced. The smallest global error is obtained with the coarse mesh of $256 \times 64$ elements. Another important parameter to observe is the CPU time. Simulations with the coarse mesh of $16 \times 4$, $32 \times 8$, $64 \times 16$, $128 \times 32$ and $256 \times 64$ elements have taken respectively 26, 8, 4, 15 and 117 s, while the benchmark MFEM provides a CPU time of 2900 s. Therefore, the best multiscale mesh arrangement is the one with the fine mesh of $512 \times 128$ elements and the coarse mesh of $256 \times 64$ elements.

![Figure 4: Global error for the velocity and pressure fields.](image)

Source: Author (2020).

5.2 Parallel Algorithm

The goal of this paper is to evaluate the performance of the parallel MuMM in simulations of the flow problem in highly heterogeneous porous media. The simulations are performed using the parallel MuMM with the fine mesh of $512 \times 128$ elements and the coarse mesh of $256 \times 64$ elements, considering distinct strengths of heterogeneity in permeability.

It is known that the strength of heterogeneity in permeability brings difficulties for the simulation of the flow problem. In this regard, the global error of the velocity and pressure fields can be analyzed in Fig. 5. The global error is greater as the strength of heterogeneity in permeability increases, for both the velocity and pressure fields.

Performance metrics for the parallel MuMM are presented for distinct strength of heterogeneity in permeability. Table 2 shows the CPU time and speedup for the following strength of heterogeneity: 0.1, 1.0, 2.0 and 4.0. The longest CPU time is found with the strength
of heterogeneity equal to 4.0; in the most threads, the greatest speedup is found with 2.0. As the number of threads is increased, the CPU time becomes smaller and the speedup, greater. The corresponding extreme values appear in bold face.

<table>
<thead>
<tr>
<th>Thread</th>
<th>Strength of heterogeneity</th>
<th>Time (s)</th>
<th>Speedup</th>
<th>Time (s)</th>
<th>Speedup</th>
<th>Time (s)</th>
<th>Speedup</th>
<th>Time (s)</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>1.0</td>
<td>115</td>
<td>-</td>
<td>126</td>
<td>-</td>
<td>134</td>
<td>-</td>
<td>166</td>
<td>-</td>
</tr>
<tr>
<td>1.0</td>
<td>2.0</td>
<td>79</td>
<td>1.6</td>
<td>111</td>
<td>1.2</td>
<td>149</td>
<td>1.1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.0</td>
<td>2.0</td>
<td>72</td>
<td>1.8</td>
<td>75</td>
<td>1.8</td>
<td>95</td>
<td>1.7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.0</td>
<td>2.0</td>
<td>69</td>
<td>1.8</td>
<td>60</td>
<td>2.3</td>
<td>81</td>
<td>2.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.0</td>
<td>2.0</td>
<td>63</td>
<td>2.0</td>
<td>58</td>
<td>2.3</td>
<td>81</td>
<td>2.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.0</td>
<td>2.0</td>
<td>59</td>
<td>2.1</td>
<td>56</td>
<td>2.4</td>
<td>76</td>
<td>2.2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.0</td>
<td>2.0</td>
<td>59</td>
<td>2.1</td>
<td>55</td>
<td>2.4</td>
<td>78</td>
<td>2.1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.0</td>
<td>2.0</td>
<td>58</td>
<td>2.3</td>
<td>75</td>
<td>2.2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

From the information shown in Table 1, we can consider that the fraction of the CPU time available for parallel computations is about 13%. According to the Amdahl law (AMDAHL, 1967), the maximum speedup would be 4.2 with parallel implementation with 8 threads and strength of heterogeneity of 4.0. This result differs from 2.2 because a greater fraction of the CPU time should be actually available for sequential computations.

The pressure, permeability, and error fields with strength of permeability heterogeneity of 4.0 are illustrated in Fig. 6. When comparing the pressure fields, the parallel \( \text{MuMM} \)

**Figure 5: Global error with distinct strengths of heterogeneity.**

**Table 2: Computational performance for distinct strengths of heterogeneity.**
solver with fine mesh of $512 \times 128$ elements and coarse mesh of $256 \times 64$ elements produces a numerical approximation with good quality in relation to the benchmark $MFEM$. The greatest local errors can be noted in transversal zones to the flow that correspond to fronts of reverse gradient of permeability in the porous medium.

Figure 6: Computed fields with strength of heterogeneity of 4.0.

(a) $MFEM$ pressure field  
(b) $MuMM$ pressure field  
(c) Permeability field  
(d) Relative error field

Source: Author (2020).
6 CONCLUSIONS

In this paper, a single-phase, incompressible flow in heterogeneous porous media is simulated. In searching for computational efficiency, the $MuMM$ multiscale method for solving the problem is employed in a parallel version using the $OpenMP$ interface. Numerical experiments are performed in order to check the advantages of the parallel algorithm.

The best multiscale mesh arrangement is that with the fine mesh of $512 \times 128$ elements and the coarse mesh of $256 \times 64$ elements. This arrangement allows the $MuMM$ solver to compute accurately average values on the interface of coarse elements.

It is evident that a prior analysis using the Performance Profiler of the Visual Studio is applicable for finding the loops that are time consuming in the $MuMM$ solver. This computational task reveals parts of the solver that should be implemented in parallel, as well as the better directives suited for it.

The parallel algorithm of the $MuMM$ solver presents a reasonable computational performance in good agreement with the benchmark $MFEM$, despite the relative global error produced in numerical simulations. The results show that is plausible to decrease efficiently the CPU time consumption of the parallel algorithm using the $OpenMP$ interface with a large number of threads.

ACKNOWLEDGMENTS

The authors acknowledge the agencies CAPES and FAPERJ for the scholarships granted to Sérgio Silva e Hanna Arimatéia, respectively.

REFERENCES


Edição especial - XXII ENMC (Encontro Nacional de Modelagem Computacional) e X ECTM (Encontro de Ciência e Tecnologia dos Materiais)

Enviado em: 29 jun. 2020
Aceito em: 10 ago. 2020
Editor responsável: Rafael Alves Bonfim de Queiroz

Revista Mundi, Engenharia e Gestão, Paranaguá, PR, v. 5, n. 6, p. 284-01, 284-14, 2020 DOI: 10.21575/25254782rmetg2020vol5n61333