INTERNATIONAL JOURNAL OF
NUMERICAL ANALYSIS AND MODELING
Volume 16, Number 6, Pages 873–890

APPLICATION OF PARALLEL AGGREGATION-BASED MULTIGRID TO HIGH RESOLUTION SUBSURFACE FLOW SIMULATIONS

MENGHUO CHEN AND SHUYU SUN*

Abstract. In this paper we assess the parallel efficiency issues for simulating single phase subsurface flow in porous media, where the permeability tensor contains anisotropy rotated with certain angles or severe discontinuity. Space variables are discretized using multi-points flux approximations and the pressure equations are solved by aggregation-based algebraic multigrid method. The involved issues include the domain decomposition of space discretization and coarsening, smoothing, the coarsest grid solving of multigrid solving steps. Numerical experiments exhibit that the convergence of the multigrid algorithm suffers from the parallel implementation. The linear system at the coarsest grid is solved and by various iterative methods and the experimental results show that the parallel efficiency is less attenuated when sparse approximate inverse preconditioning conjugate gradient is used.

Key words. Parallel computation, porous media flow, multi-points flux approximations, algebraic mulitgrid.

1. Introduction

Modeling of subsurface flow processes is important for many applications, for example, groundwater resource management, CO_2 sequestration, and petroleum production. In all the cases, fluids flow through medium containing pores (voids). A porous media is most often characterized by porosity and the media's ability to transmit fluids is measured by a quantity called permeability. To describe the physics of the fluid flow, the Darcy's law is used to establish the partial differential equations, where the geometrical complexities of porous media is replaced by one parameter relates to pressure gradient vector and the fluid flux vector. As a consequence of the different geologic processes over geologic time-scales, however, there are cases such that the rock formations is geometric complicate in a way the hydraulic properties of media are mostly heterogeneous and anisotropic, where the flux vector does not coincide the hydraulic gradient vector.

The appearance of anisotropy in permeability which is not aligned with the direction of principal axes imposes challenges on the development of reservoir simulator and the stencil arising from the standard two-point flux approximation (TPFA) fails to account for the fact that pressure gradient in one direction can cause flow in other directions as well. This motivates the development of multi-point flux approximation (MPFA) [1, 2], which is designed based on the finite volume formulation. MPFA introduces the surface midpoints through the interaction region to ensure the pressure and flux continuities and the flux is calculated using informations involving several points as opposed to only two points. This allows the application of MPFA methods on general nonorthogonal grids as well as for general orientation of

Received by the editors December 1, 2017 and, in revised form, September 27, 2018.
2000 *Mathematics Subject Classification.* 66M66, 65Y05.
*Corresponding author.

the principal directions of the permeability tensor. In the work of Aavatsmark et al. [1], MPFA is introduced with two subclasses, i.e., MPFA O- and U- methods, based on the choice of continuity conditions. In [3] Aavatsmark discussed the discretization and implementation of MPFA-O method with subsurface midpoint as continuity points on quadrilateral grids. In addition, Edwards and Rogers [2] have demonstrated the existence of an error produced by the classical five-point stencil scheme (TPFA) due to the ignorance of the off-diagonal elements in the tensor. The off-diagonal elements of the tensor equation could not be ignored since these give a strong impact on the variation of the pressure field. Besides the mentioned subclasses, there are other types of MPFA such as MPFA L-method [4] and MPFA Z-method [5].

The linear systems arising from MPFA methods can be solved efficiently by direct methods (Gauss eliminations) when the number of space cells is small [6, 7]. In recent years, however, the study of flow phenomena in multiple scales (e.g., from pore scale to Darcy scale) and in fractured porous media have drawn significant attention [8, 9, 10, 11]. In these research the mesh with high resolution is needed in order to resolve wide range of length scale of interest and solving the associated linear system using direct method is not favourable. As the resulting matrix from MPFA being sparse, iterative methods turn out to be better options. Among the iterative methods, multigrid methods are well known for being fastest numerical methods for solving linear systems arising from discretization of elliptic partial differential equations (PDEs) [12]. For large classes of problems it can be shown that the convergence rate of multigrid method will not deteriorate as the mesh size increases and the total work to solve a linear system with N variables is $O(N)$ if a fixed level of accuracy is needed [13, 14].

In this paper, we incorporate an aggregation-based algebraic multigrid (AGMG) method [15] in solving subsurface flow equations. The cases studied include the TPFA discretized pressure equations in single-phase flow where the permeability contains discontinuity with large jump and rotated anisotropy. On the other hand, we consider the linear system arising from MPFA discretization. Furthermore, the parallelization of the solution algorithm is implemented and carried out in massively parallel simulations. Various issues and difficulties are addressed and discussed. The remaining part of the paper is organized as follows. In Section 2 we state the mathematical model of fluid flow in porous media, MPFA method and experimenting field approach. In Section 3 we discuss several issues for parallelizing multigrid methods. In Section 4 the setup of numerical experiments and related facility are discussed. Results of several numerical experiments are presented in Section 5.

2. Subsurface flow in anisotropic porous media

2.1. Governing equations. The governing equations of fluid flow in porous media are given by the combination of physical principles: the conservation of mass and Darcy's law. The principle of mass conservation assumes the mass inflow and outflow are equal when fluid flow crosses a certain region. Thus the mass conservation equation is given by

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

where ϕ is the porosity, ρ is the density of the fluid. $\mathbf{q}$ is the sink or source. $\nabla\cdot$ is the divergence operator and $\mathbf{u}$ is the velocity obeying the Darcy's law, which is defined as

$$\mathbf{u} = -\frac{\mathbf{K}}{\mu}(\nabla p - \rho g \nabla \hat{z}). \tag{2}$$

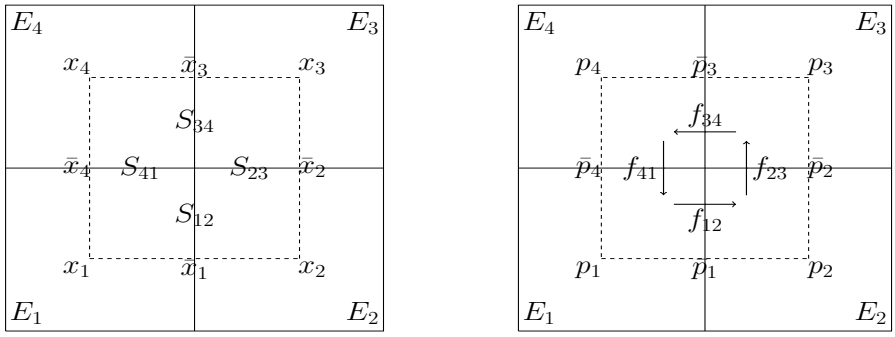
In equation (2), $\mathbf{K}$ is the full permeability tensor, p is the pressure, μ is the fluid viscosity, g is the gravitational acceleration, $\hat{z}$ is the depth, and ∇ is the gradient operator. In the case of 2D space, $\nabla = \left(\frac{\partial}{\partial x}, \frac{\partial}{\partial y}\right)^T$, $\mathbf{K}$ is represented as a second order tensor by

$$\mathbf{K} = \begin{bmatrix} K_{xx} & K_{xy} \\ K_{yx} & K_{yy} \end{bmatrix}. \tag{3}$$

2.2. Multipoint flux approximation. Consider the four quadrilateral cells with a common vertex in Figure 1a. Denote the cells by E_i, the cell centers by x_k, and the edge midpoints by $\bar{x}_k$, where $i, k = 1, 2, 3, 4$. Lines are drawn between the cell centres and the midpoints of the edges (shown as dashed lines in the figure). These lines bound an area around each vertex which is called an interaction volume. As seen from the figure, there are four half cell edges (solid lines) in the interaction volume. To discretize the PDEs (1) - (2) using MPFA, we first compute the flux of potential of a phase α through an half cell edge, say, S in an interaction volume:

$$f = -\int_S (\mathbf{K}\nabla\Phi)\cdot\hat{\mathbf{n}}\,\mathrm{d}S \tag{4}$$

where ϕ is the phase potential (which is pressure in our case), $\mathbf{K}$ is the transmissibility tensor (permeability in our example), and $\hat{\mathbf{n}}$ is the unit normal vector to the surface. Note that for convenience we drop the phase index α from (4). Assume



(a) location of points (b) pressure and fluxes

Figure 1. Cells and interaction volume.

that the potential is a continuous function. The gradients of the potential $\nabla\Phi$ are approximated by the difference between the potential values at the cell centres and that at the edge centres. Now the potential considered here is pressure and we denote it by p. To preserve local conservation of fluxes it is required that the flux

leaving one control volume is equal to that entering the next one. The approximated value of flux in (4) through half cell edge S_{12} and through S_{41} (see Figure 1) are

$$\begin{aligned} f_{12} &\approx -K_1^{xx}(\bar{p}_1 - p_1) - K_1^{xy}(\bar{p}_4 - p_1) \\ &= -K_2^{xx}(p_2 - \bar{p}_1) - K_2^{xy}(\bar{p}_2 - p_2), \end{aligned}$$

$$\begin{aligned} f_{41} &\approx -K_1^{yy}(\bar{p}_4 - p_1) - K_1^{yx}(\bar{p}_1 - p_1) \\ &= -K_4^{yy}(p_4 - \bar{p}_4) - K_4^{yx}(\bar{p}_3 - p_4), \end{aligned}$$

where $K_i, i = 1, 2, 3, 4$ is the transmissibility coefficient at the center of the cell, E_i, p_i and $\bar{p}_i$ are the pressure at the cell centres and edge midpoints, respectively. The fluxes f_{23} and f_{34} are obtained in a similar fashion. Each flux through the half cell edge involves two adjacent cells to be considered. From each interaction region, we would obtain four systems of equations, which can be solved locally for the transmissibility matrix $\mathbf{T}$, such that

$$\mathbf{Tp} = \mathbf{f}. \tag{5}$$

Consider the steady state mass conservation equation for the flow of an incompressible fluid in porous media with source/sink term $\mathbf{q}$. The equation can be written simply as

$$\nabla \cdot \mathbf{u} = \mathbf{q}. \tag{6}$$

If the gravity is lumped into the pressure, by Darcy's law we obtain the velocity as

$$\mathbf{u} = -\frac{\mathbf{K}}{\mu}\nabla p. \tag{7}$$

Suppose that the pressure field is known, then the velocity field could be obtained using the Darcy's law and such $\mathbf{u}$ should satisfy Equation (6). However, if the pressure is not the correct one, the equality does not hold in Equation (6). Define the residual $\mathbf{R}$ as the difference between the true value and the calculated one using the guessed pressure field (denoted by $\tilde{p}$):

$$\mathbf{R} = \mathbf{q} - (\nabla \cdot \mathbf{u})_{\tilde{p}}, \tag{8}$$

we will use $\mathbf{R}$ to refine the velocity field. Substituting Darcy's law into the conservation of mass law, one obtains

$$\nabla \cdot \mathbf{u} = \nabla \cdot (-\frac{\mathbf{K}}{\mu}\nabla p). \tag{9}$$

In matrix form this equation may be represented as

$$\mathbf{Ap} = [\nabla \cdot \mathbf{u}]. \tag{10}$$

Therefore, for the i^{th} testing pressure field (denoted by $\tilde{p}^i$) we have

$$\mathbf{Ap}^i = \mathbf{q} - [\nabla \cdot \mathbf{u}]_{\tilde{p}^i}. \tag{11}$$

Now if $\tilde{p}^i$ is chosen to be 1 in the ith cell (ith entry of vector $\mathbf{p}^i$) and zeros elsewhere, one could obtain the entries of the ith column of the coefficient matrix $\mathbf{A}$ and likewise until the matrix is populated. In addition, the boundary condition has to be incorporated as shown in [6] using additional pressure field where the pressure in all the cells is zero. Once $\mathbf{A}$ is constructed, we obtain the correct pressure field through equation (10). Finally the correct velocity based on the correct pressure

can be calculated by algebraic equations arising from the MPFA discretization of equation (7).

2.3. Experimenting pressure field. To calculate the entries of matrix **A** in equation (11), Sun et al. [6] propose equation type approach which generate the matrix automatically within the solver routine by looping over $m \times n + 1$ pressure fields, where m and n are the number of segments in x and y directions in 2D space. In this approach the computational work is clearly cumbersome. In the work [7], Negara et al. notice that the cell where the pressure is one affects the flux in only the neighbouring four cells. This implies that one could design the testing pressure field in which the cells where the pressure is one alternate every two cells. In other words, the number of generations of the testing pressure fields can be reduced from m by n times to only nine times, which clearly reduce significantly the time required to construct the matrix of coefficients.

The linear system in (10) can be solved by direct methods (Gauss elimination) for small gird size [6, 7]. For solutions on high resolution meshes, on the other hand, iterative methods are better alternative as the matrix **A** in equation (10) is sparse. Furthermore, computations in iterative methods are less sequential, with sometimes few global communications. Among iterative methods we choose AGMG method as our linear system solver for solving the pressure field.

3. Algebraic Mulitgrid and Its Parallelization

1. Pre-smooth $x_1 = S_1(x_0, b)$
2. Residual $b_1 = b - Ax_1$
3. Restriction $\tilde{b}_1 = Rb_1$
4. Next level $\tilde{A}\tilde{x}_2 \approx \tilde{b}_1$
5. Prolongation $x_2 = P\tilde{x}_2$
6. Correction $x_3 = x_1 + x_2$
7. Post-smooth $x_0 = S_2(x_3, b)$

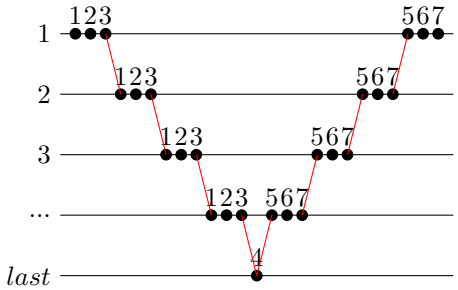


Figure 2. Multigrid algorithm (left) and illustration of V-cycle (right). S_1 , S_2, P and R are pre-smoothing, post-smoothing restricting and prolongation operators, respectively.

The principle of multigrid methods can be explained as follows: consider the linear system $Au = b$ on a fine grid, where A represents the matrix and b the right-hand side vector, and u is the solution vector that one seeks. In iterative algorithms we let $u^{(i)}$ denote the approximate solution to the linear system at the ith iteration and decompose the corresponding error $e^{(i)} = u^{(i)} - u$ into its Fourier components. Classical relaxation schemes, such as weighted Jacobi or Gauss-Seidel methods, can quickly damp the high frequency components of the error. For this reason these methods are called smoothers. When the error is dominated by low-frequency components, however, the further smoothing iterations hardly reduce the error. To solve this, the multigrid methods use coarse grid correction, that is, the low-frequency components are projected to a sufficiently small grid (see Fig. 2). On the coarser level few (one or two) relaxations (step 2) can effectively reduce

the error because the smoother components projected from the fine level appear more oscillatory. The smoothing-restriction procedure can be carried out until the coarsest grid is reached where the corresponding residual equation is inexpensive to solve directly (step 4). The coarsest grid solution is then interpolated back to the finer levels where further relaxation sweeps may be applied (step 4).

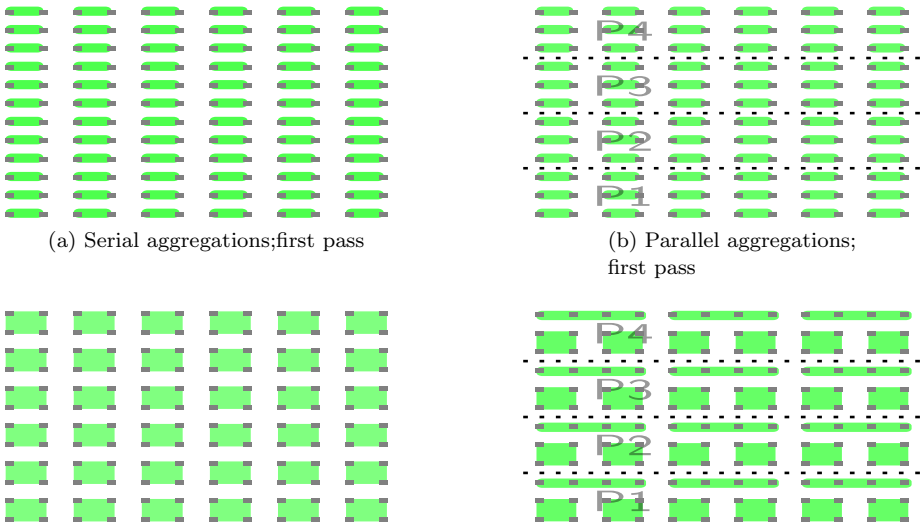
Typically multigrid algorithms involve two phases of computation. Firstly the grid hierarchy is determined in the setup phase. The information for determining the grids on coarser levels can be based on geometric location, features of PDEs (geometric multigrid) or purely the matrix entries of the linear system (algebraic multigrid). Consequently a grid hierarchy is constructed. In the second phase cycle strategies (V-, W- or K-cycles [17]) are iterated on the resulted grid hierarchy until the solution converge, where the major costs are smoothing and restriction/prolongation.

In AGMG the unknowns (say N unknowns) are subdivided into (say N_c) disjoint small groups (aggregates) which represent unknowns on next coarser level. In each aggregate, the prolongation values are assumed to be uniform, that is, functions are piecewise constant. Piecewise constant prolongation may attenuate the convergence property of the multigrid and a remedy is proposed in [17], in which K-cycle instead of V-cycle is used in cycle strategy. In K-cycle the approximate solution $\tilde{x}_2$ in Fig. 2 is obtained by one or two multigrid preconditioned Krylov subspace iterations (preconditioned conjugate gradient or GMRES), where the multigrid preconditioner is the K-cycle implementation on the next coarser level. For more details about K-cycle strategy, see [15, 17].

For parallel multigrid the rows (unknowns) of the matrix are distributed in processes. Coarsening algorithms perform grid coarsening on local rows and construct the grid hierarchy including connections between rows on different processes. Matrix-vector multiplications in smoothing use the connection information to obtain the correct results. This implementation raises issues regarding the impacts on the convergence of the multigrid.

3.1. Parallel Coarsening. The coarsening algorithm accesses the matrix only row by row. It treats the local matrix as if the non-local variables have been marked and excluded from the aggregation process at the start of the coarsening. Hence each task computes locally the coarsening scheme for local variables. This simple approach for parallel coarsening may create different aggregates from that of serial coarsening. For example, consider an N by N matrix A of finite difference discretization of a 2D Model problem with 5-point stencil. Figure 3a and 3b show the aggregates after the first pass of the coarsening algorithm of the code AGMG [18]. While the formed aggregates are the same after the first pass, they differ after the second pass, as can be seen in Figure 3c and 3d. There are aggregates with "**bad quality**" which are forced to be formed near the boundaries (internal) between the subdomains (processors), such as line aggregates which are not appropriate when there is no anisotropy. These bad aggregates may affect the convergence.

3.2. Parallel Smoothing. The solution phase involves the following steps: smoothing, computing residuals and restriction/prolongation. In general, smoothing involves inversion of a matrix albeit a simple one. For weighted Jacobi relaxation, the inversion of diagonal entries can be carried out independently for each row and is easy to solve in parallel. For Gauss-Seidel relaxation, however, the



(a) Serial aggregations;first pass

(b) Parallel aggregations; first pass

(c) Serial aggregations; second pass

(d) Parallel aggregations; second pass

Figure 3. Pairwise aggregation for serial and parallel cases.

backward/forward substitutions needed for inverting triangular matrices are well known to be inherently sequential. In order to perform backward/forward substitutions in parallel, the processors ignore the matrix entries indicating connections to neighbouring processors and all processors solve their part of the residual equation simultaneously. Consequently the global smoother matrix is block diagonal:

$$\begin{bmatrix} \mathbf{B}_1 & 0 & 0 & 0 \\ 0 & \mathbf{B}_2 & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & \mathbf{B}_p \end{bmatrix}$$

where $\mathbf{B}_i$ is the lower or upper triangular part of the local matrix A_i ($\lceil \frac{n}{p} \rceil \times n$), $i = 1, ..., p$. The convergence may suffer from this modification of the smoothers. In order to see its impact on the convergence of multigrid, we perform numerical experiments which use the code in [18] to solve the 2D Model problem with Dirichlet boundary condition on a 8000×8000 mesh. In the experiments V-cycle instead of K-cycle strategy is chosen and the multigrid is used as preconditioner for conjugate gradient (CG). The experimental results are shown in Table 1. As seen from the table, the number of multigrid V-cycle preconditioned CG iteration increases significantly. This demonstrates the impact of the parallel implementation for multigrid on the parallel efficiency. On the other hand, if K-cycle is used the number of multigrid PCG cycles does not change significantly. As we will see later, however, for some problems even the use of K-cycle can not recover the degraded convergence caused by parallelization.

3.3. Solving the Coarsest Grid Problem. Solving the coarsest grid problem in the multigrid cycle may be critical to the performance of the parallel multigrid

Table 1. Effect of parallelization on the convergence of AGMG, V-cycle preconditioned CG.

No. of processors	# level	Time setup + solution	No. of multigrid PCG iterations
1	8	53.2+354	56
4	8	13.9+93.1	76
8	8	7.45+58.2	77
16	8	4.16+29.7	78
25	8	2.87+20.9	107
64	8	2.04+8.9	110

method, as mentioned in [19]. Two factors are concerned here: the sparsity of the coarsest grid matrix and the number of coarsest grid visit. For AGMG grid hierarchy, we observed that the average number of entries per row at the coarsest level increases slightly ($5 \rightarrow 6$ for 5-point stencil or $9 \rightarrow 11$ for 9-point stencil). This can attribute to the simple prolongation operator [15]. On other hand the coarsest grid visit in AGMG is 2^{l-2} (l represents the number of grid levels) in all problems studied in this research. Because of the above observations, we use preconditioned conjugate gradient (PCG), an iterative methods, to solve the coarsest grid problem in parallel AGMG. Two chosen preconditioners are diagonal preconditioning (DPCG) and sparse approximate inverse preconditioning (SAIPCG).

3.4. Sparse Approximate Inverse (SAI) and Diagonal preconditioner. When a preconditioner M is applied in the conjugate gradient (CG) algorithm, one needs to solve the equation $Mz_k = r_k$ in each CG iteration, so that the modified residual z_k can be obtained from the residual r_k at iteration k. Either we solve the equation directly, for example by backward or forward substitution if M is triangular, or if M^{-1} is available we can compute $z_k = M^{-1}r_k$. Many popular general-purpose preconditioners, such as those based on incomplete factorization, are fairly robust and result in good convergence rates, but are highly sequential and it is difficult to implement them efficiently on parallel computers. On the other hand, sparse approximate inverse [20] approximates the matrix inverse while retains certain degree of sparsity. This allows us to compute $z_k = M^{-1}r_k$ in terms of matrix-vector multiplication, which is easier to be parallelized.

Starting with a symmetric positive definite matrix A, we construct the its approximate inverse as follows: the inverse A^{-1} of A is approximated by the factorization G^TG, where G is a sparse lower triangular matrix approximating the inverse of the lower triangular Cholesky factor, L, of A [21]. G is obtained so that $\|I - GL\|_F$ is minimized, where $\|\cdot\|_F$ denotes the Frobenius norm, subject to some sparsity constraint. Minimizing $\|I - GL\|_F$ can be accomplished without knowing L by solving the normal equations :

$$(GLL^T)_{ij} = (L^T)_{ij}, \qquad (i,j) \in S_L, \tag{12}$$

where S_L is a lower-triangular nonzero pattern for G. Equation (12) can be replaced by

$$(\hat{G}A)_{ij} = I_{ij}, \qquad (i,j) \in S_L, \tag{13}$$

where $\hat{G} = D^{-1}G$ and D is the diagonal of L. For simplicity, we use the sparsity pattern of the lower triangle of A to construct the approximate inverse. The constrained minimization problem decouples into n independent linear least squares

problems (one for each row of G). The number of unknowns for each problem is equal to the number of nonzero allowed in each row of G. This immediately follows from the identity:

$$\|I - GL\|_F^2 = \sum_{i=1}^{n} \|I(i,:) - G(i,:)L\|_2^2. \tag{14}$$

These linear least squares problems can be solved independently for each row $G(i,:)$ directly or iteratively, therefore the computation can be parallelized.

Besides sparse approximate inverse preconditioner, we also employed the diagonal preconditioning for comparison. The diagonal preconditioning CG (DPCG) is simple and only one matrix-vector multiplication is needed in preconditioning step, while in SAI preconditioned CG (SAIPCG) the preconditioner has more than one entry per row. However, SAIPCG generally converges faster than DPCG. In the following sections, we will compare their performance as the coarsest grid solver. The CG iterations stop when the relative residual is below the given tolerance. The number of multigrid levels and the tolerance are tuned so that the optimal performance is obtained.

4. Numerical Experiments Setup

In this work, we consider examples which are related subsurface flow in porous media: pressure equations with jump coefficients, equations with rotated anisotropy where the direction of anisotropy form a certain angle with x-coordinate. It is noticed that the PDEs in these examples contain coefficients with various complexity, which are different from the Poisson-like equations considered in [25]. The investigation of the efficiency for massively parallel AGMG therein employed couple examples which are different by the given right hand side (f in equation 1.2, [25]).

All numerical experiments were carried out on Shaheen I, a 16-rack Blue Gene/P consisting of 16384 computing nodes. Each node is equipped with four 32-bit, 850 Mhz PowerPC 450 cores and 4GB DDR memory. In aggregate, Shaheen has 65,536 compute cores and 64TB of memory. The Blue Gene/P architecture provides a three-dimensional point-to-point Blue Gene/P torus network for general-purpose IPC. Each torus link can transmit up to 425 MB per second in each direction, for a total of 5.1 GB per second bidirectional bandwidth per node.

4.1. The Tested Linear Systems.

Problem JUMP2D: The first case is the linear system arising from the TPFA discretization of PDE. Figure 4 shows the geometry of the permeability coefficients of the test problem JUMP2D on the $[0,1] \times [0,1]$ domain. The PDE coefficients contain anisotropies along the horizontal and vertical axes, as well as several orders of magnitude of jump. Such PDE may occur when one solve the subsurface flow in fractured porous media. This problem is similar to the problem in [22], where we consider a larger coefficient jump (3 orders of magnitude) and apply Dirichlet boundary condition on all four sides of the boundary.

$$-\frac{\partial}{\partial x}(a\frac{\partial p}{\partial x}) - \frac{\partial}{\partial y}(b\frac{\partial p}{\partial y}) = f \qquad \text{in} \qquad \Omega = [0,1] \times [0,1] \tag{15}$$

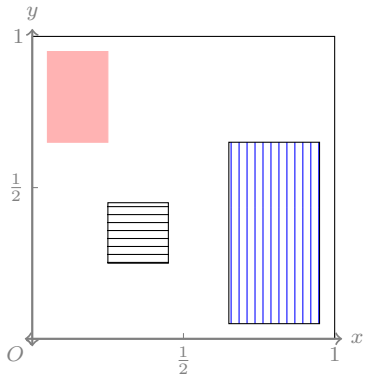


Figure 4. JUMP2D.

with coefficients given by

$$
\begin{cases}
a = 1, & b = 1000, & f = 0, & \text{in } (0.65, 0.95) \times (0.05, 0.65), \\
a = 1000, & b = 1, & f = 0, & \text{in } (0.25, 0.45) \times (0.25, 0.45), \\
a = 1000, & b = 1000, & f = 1, & \text{in } (0.05, 0.25) \times (0.65, 0.95), \\
a = 1, & b = 1, & f = 0, & \text{in elsewhere},
\end{cases} \tag{16}
$$

and Dirichlet boundary condition:

$$
p = 0, \text{ on } \partial\Omega. \tag{17}
$$

Problem ROTANISO2D : For the second case we test the linear system arising from the TPFA discretization of the pressure equation with rotated anisotropy:

$$
\begin{aligned}
&-(\epsilon \cos^2\theta + \sin^2\theta)\frac{\partial^2 p}{\partial x^2} - 2(1-\epsilon)\cos\theta\sin\theta\frac{\partial^2 p}{\partial y \partial x} - (\cos^2\theta + \epsilon\sin^2\theta)\frac{\partial^2 p}{\partial y^2} = 1 \\
&\Omega = [0,1] \times [0,1] \\
&p = 0, \qquad \text{on} \qquad \partial\Omega.
\end{aligned} \tag{18}
$$

where the anisotropy ratio ϵ is 0.001. The smaller the anisotropy ratio is, the much stronger anisotropy exerts on the direction of subsurface flow. The anisotropy angles θ considered for comparison are $\pi/12$, $\pi/6$ and $\pi/4$. Problems of this sort can be a challenge for AGMG. If the direction of anisotropy is not aligned with the grid lines, the convergence rate of AGMG is severely h (grid space) dependent. The analysis in [23] exhibits that if the chosen aggregates can align (as much as possible) with the direction of anisotropy then h and ϵ-independent convergence can be obtained with a two-grid method. The work also propose a modified aggregation strategy to improve the convergence of AGMG and the numerical experiments demonstrate the improvement in the serial code. For the best result we apply the proposed aggregation strategy in [23] in this example.

Problem MPFA : The last example tested is the linear system in equation (10), which arises from the MPFA discretization of the PDEs (1)-(3) on staggered grid. In other words, the velocity variables are edge-centred and the pressure variables are cell centred, see Figure 5. For the boundary conditions, we impose velocity boundary condition of 2×10^{-6} m/s on the left edge of the domain. On the right

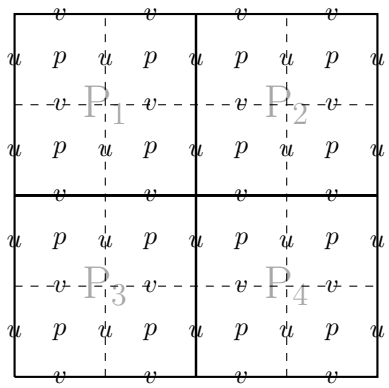


Figure 5. Staggered grid divided by 4 processes.

hand boundary the pressure is set to 1 atmosphere (1.01325 bar). No flow boundary condition is assumed on the top and bottom of the domain.

As described in section 2.2 - 2.3, the algorithm first calculates the matrix entries of **A** in equation (10) by solving local problems associated with experimenting fields, then the resulting linear system is solved by AGMG. The processors communicate when updating the edge-center variables. The MPFA discretization on each processor are simultaneously carried out and the constructed matrix **A** are fed to the AGMG solver to obtain the pressure field. Besides the performance of multigrid solver, we also observe the parallel efficiency of the MPFA discretization.

4.2. Parameters. In this research we investigate the weak scaling efficiency. In all experiments, the 2D space domain is divided into $q \times q = q^2 = p$ blocks, where q is an even power of 2. The number of variables per block (task) is 40000. Each timing result is the average of multiple experiments. For the coarsest grid solving the PCG iterations stop when the 2-norm of the relative residual is ≤ 0.90, a heuristic value we found to be optimal. All the parallel timing results are compared to the result from using single processor in order to calculate the parallel efficiency. The coarsest grid problem in the single process case is solved by the direct solver from LAPACK library.

5. Results and Discussions

I. Problem JUMP2D: Table 2 shows the timing results for the performance of parallel AGMG, using diagonal preconditioned CG (DPCG) and sparse approximate inverse preconditioned CG (SAIPCG) as the coarsest grid solver. At the end of each timing result in the tables, the number of AGMG iterations is parenthesized.

As seen from the table, SAIPCG performs better than DPCG does, although the difference is not significant. Furthermore, the cost for constructing sparse approximate inverse is not significant, as the number of normal equations to solve is very few at the coarsest grid. We also notice that in most of the experiments the total number of SAIPCG iterations is roughly twice the number of DPCG iterations at

Table 2. Timing results: JUMP2D.

n $\times 10^4$	#p	# level	Time (setup+solve) (# it) DPCG	Time (setup+solve) (# it) SAIPCG
4	1	6	0.42+2.95(21)	0.42+2.95 (21)
8	16	7	0.48+3.17(21)	0.48+3.09 (21)
256	64	8	0.48+3.24(21)	0.49+3.13 (21)
1024	256	8	0.49+3.30 (21)	0.51+3.20 (21)
4096	1024	8	0.48+3.37(22)	0.51+3.30 (21)
16384	4096	8	0.53+3.91(22)	0.55+3.71 (21)
65536	16384	8	0.57+4.29 (22)	0.60+4.10 (21)

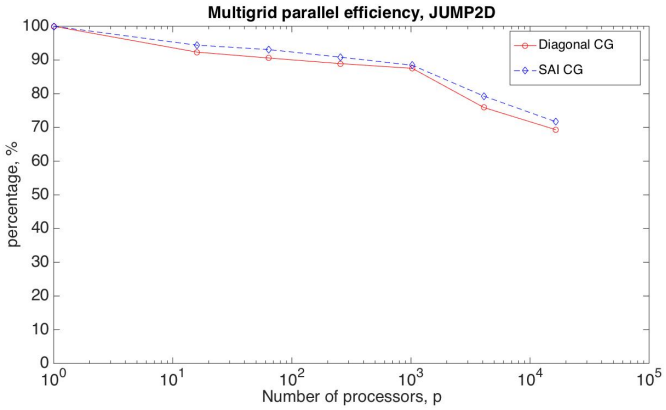


Figure 6. Parallel efficiencies for using DPCG and using SAIPCG as coarsest grid solver. JUMP2D.

the coarsest grid. However, the extra matrix-vector multiplications and the associated communications in SAIPCG weaken its advantage of faster convergence over DPCG. The parallel efficiency is around 70% in the worst case (see Figure 6), and the convergence is almost grid size independent. The results show that the application of parallel AGMG on pressure equation with coefficients containing strong discontinuity looks promising.

II. Problem ROTANISO2D : In this case we carry out experiments for solving equation (18). Three angles are considered: $\pi/12$, $\pi/6$ and $\pi/4$. As mentioned in section 4.1, the modified aggregation strategy proposed in [23] is used to reduce the grid-size dependent convergence.

From Table 3 - 5 we see severe grid size dependency on the convergence. For $\theta = \pi/4$ the dependency is relatively mild and the parallel efficiency is roughly 40% for the largest grid. For the scenario $\theta = \pi/12$ the convergence is too slow for large grid and the results are not shown here. This problem has been a challenge for AGMG and the unimpressive performance is expectable. Similar to the case **JUMP2D**, SAIPCG performs better than DPCG does as the coarsest grid solver.

Table 3. Timing results: ROTANISO2D, $\theta = \pi/12$.

n $\times 10^4$	#p	# level	Time (setup+solve) (# it) DPCG	Time (setup+solve) (# it) SAIPCG
4	1	6	1.61+12.4(70)	1.61+12.4(70)
8	16	7	1.9+22.5(89)	1.93 + 20.5(88)
256	64	8	1.92+25.6(94)	1.93 +24.3 (93)
1024	256	8	1.94+40.5(147)	1.94+37.1(151)
4096	1024	8	2.16 + 86.1 (292)	2.19+ 64.3 (291)

Table 4. Timing results: ROTANISO2D, $\theta = \pi/6$.

n $\times 10^4$	#p	# level	Time (setup+solve) (# it) DPCG	Time (setup+solve) (# it) SAIPCG
4	1	6	2.01+17.1(82)	2.01+17.1(82)
8	16	7	2.02+20.5(90)	2.03 +19.5(91)
256	64	8	2.07+25.6(104)	2.05+22.6(104)
1024	256	8	2.16+32.4(116)	2.13+30.8(117)
4096	1024	9	2.24+36.4(132)	2.14+33.6(126)
16384	4096	9	2.33+40.3(137)	2.29+38.5(135)
65536	16384	9	2.49+46.4(143)	2.54+44.8(141)

Table 5. Timing results: ROTANISO2D, $\theta = \pi/4$.

n $\times 10^4$	#p	# level	Time (setup+solve) (# it) DPCG	Time (setup+solve) (# it) SAIPCG
4	1	6	1.98+10.1(55)	1.98+10.1(55)
8	16	7	2.03+12.2(60)	2.05+11.5(60)
256	64	8	2.09+16.6 (82)	2.10+15.7(82)
1024	256	8	2.16+19.2(90)	2.18+17.8(90)
4096	1024	9	2.20+22.4 (96)	2.24+20.6(96)
16384	4096	9	2.31+26.6(101)	2.34+24.2(100)
65536	16384	9	2.45+29.2(104)	2.44+26.5(103)

III. Problem MPFA: Similar to problem **ROTANISO2D**, we consider three anisotropy angles: $\pi/12$, $\pi/6$, $\pi/4$. The results are shown in Table 6 - 8. In the tables the execution times for the MPFA discretization mentioned in subsections 2.2 -2.3 are listed in the forth column. Note that for the case $\theta = \pi/12$, two smoothing steps at each level are needed so that the number of AGMG iterations (≤ 150) is acceptable. Furthermore, the original aggregation strategy in [22] is used here since it provides the optimal aggregates. We also observe that slow coarsening (coarsening ratio is less than 3) appears when the multigrid level is greater than 7 or 8. In this situation, the saving of computational work gained from the coarser level does not compensate the increasing number of the visiting for the coarsest level (in each W-cycle the coarsest level is visited 2^{l-2} times, where l is the number of the multigrid levels), therefore we set the maximum number of multigrid levels as 8 for $\theta = \pi/4, \pi/12$, 9 for $\theta = \pi/6$. This implementation is inspired from the

Table 6. Timing results: MPFA, $\theta = \pi/12$.

n $\times 10^4$	#p	# level	MPFA discretization	Time (setup+solve) (# it) DPCG	Time (setup+solve) (# it) SAIPCG
4	1	6	47.3	0.60+9.95(34)	0.60+9.95(34)
8	16	7	48.7	0.72+15.4(42)	0.73+15.0(42)
256	64	8	48.9	0.75+15.5(44)	0.78+15.2(43)
1024	256	8	49.7	0.74+16.1(46)	0.82+15.3(45)
4096	1024	8	50.9	0.77+16.1(47)	0.85+15.6(47)
16384	4096	8	51.0	0.95+16.2(48)	1.02+15.8(48)
65536	16384	8	51.3	1.03+17.1(48)	1.10+16.3(48)

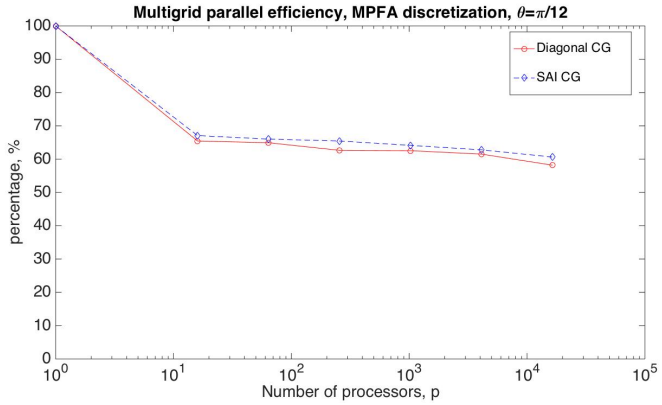


Figure 7. Multigrid parallel efficiency for solving subsurface flow in porous media with MPFA discretization, $\theta = \pi/12$.

Table 7. Timing results: MPFA, $\theta = \pi/6$.

n $\times 10^4$	#p	# level	MPFA discretization	Time (setup+solve) (# it) DPCG	Time (setup+solve) (# it) SAIPCG
4	1	6	46.5	0.65+11.9 (48)	0.65+11.9 (48)
8	16	7	49.2	0.69 +14.9 (59)	0.71+14.6 (59)
256	64	8	49.3	0.71 +15.8 (62)	0.68+15.4 (62)
1024	256	8	49.9	0.73+17.6(63)	0.75+16.9 (62)
4096	1024	9	49.8	0.93 +19.0(62)	0.95+18.3 (62)
16384	4096	9	50.1	0.96+19.2 (63)	0.98+19.0 (62)
65536	16384	9	50.1	0.97 + 20.1 (63)	1.03 +19.8 (62)

work in [25], where the slow coarsening was also observed in the tested examples therein.

In Figure 11 we also show the relation between the convergence rate of AGMG with the same total levels and the rotation angles of anisotropy. Notice that for this figure the coarsest grid problem is solved exactly since the accuracy of the solution from the coarsest grid affects the convergence of the overall AGMG algorithm significantly. As seen from the figure, the convergence rate of AGMG for the rotation angle $\theta = \pi/4$ is the best in all cases, which is expectable since for $\theta = \pi/4$ the (line) aggregates can align with the direction of the anisotropy nicely.

From the tables, parallel AGMG performs best when $\theta = \pi/4$. In that case the number of cycles does not grow significantly as the grid size increases. By investigating the aggregations, we observe that the direction of aggregations are aligned with the direction of anisotropy at all multigrid levels. For the other two cases the convergence of AGMG is grid size dependent but the dependency becomes

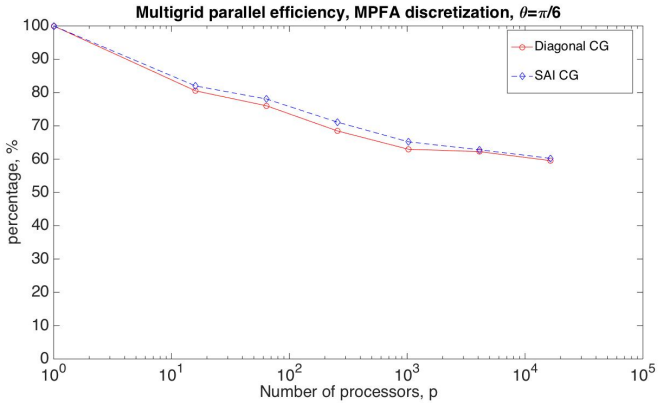


Figure 8. Multigrid parallel efficiency for solving subsurface flow in porous media with MPFA discretization, $\theta = \pi/6$.

Table 8. Timing results: MPFA, $\theta = \pi/4$.

n $\times 10^4$	#p	# level	MPFA discretization	Time (setup+solve) (# it) DPCG	Time (setup+solve) (# it) SAIPCG
4	1	6	46.7	0.69+4.20(17)	0.69+4.20 (17)
8	16	7	49.97	0.67 + 4.54(17)	0.7+4.42 (17)
256	64	8	50.1	0.67+4.67(17)	0.65+4.86(17)
1024	256	8	50.5	0.86+5.1(18)	0.80+5.35(18)
4096	1024	8	50.25	0.88+5.65(19)	0.86+5.87(20)
16384	4096	8	50.52	0.96+5.7 (19)	0.97+6.18 (20)
65536	16384	8	50.88	1.02+6.35(20)	0.96+6.60(21)

mild when the grid size exceeds certain scale($\geq$ 1024 processors). The parallel efficiency for the multigrid solving step varies from 55% to 92%, see Figures 7 - 9. It is also observed that SAIPCG is a better coarsest grid solver than DPCG.

The time for parallel MPFA discretization of the PDEs increases slightly as the number of the processors increases, and the parallel efficiency is above 90%. This is attribute to the inherent parallelism of MPFA discretization algorithm, in which the rows of the global transmissibility coefficient matrix **A** are constructed using only information from at most eight neighbouring nodes. This localized computation does not need global communications (global broadcasting). In all three cases the solution time for solving pressure field is a fraction of the MPFA discretization step, thus the parallel efficiency of the overall simulation is above 80% for the experiments in this work, which is quite satisfactory, see Figure 10.

6. Conclusion

The issues regarding the massively parallel AGMG for various examples are discussed. The impact of parallel aggregation and smoothing on the AGMG convergence is not obvious for the problem **JUMP2D** which contains discontinuous coefficient jump with several orders of magnitude. This indicates that the application of AGMG on the problem for flow in fractured media is promising. In general, the ratio of permeability between fractures and rock matrix can be 3 or 4 orders of magnitude. Furthermore, parallelized computation with good parallel efficiency facilitates the simulation for this sort of problems, which in general need high resolution meshes.

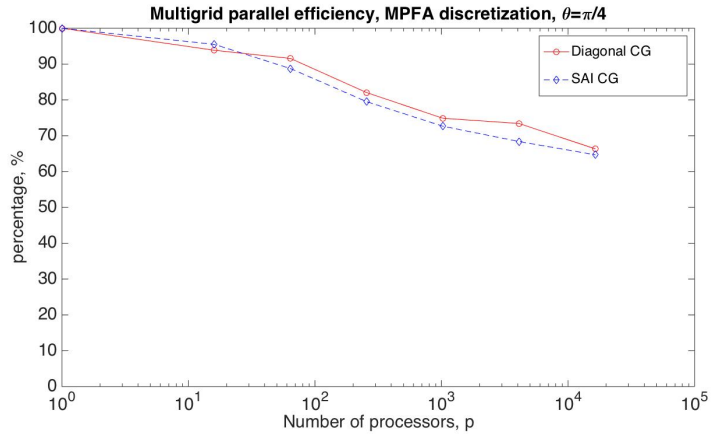


Figure 9. Multigrid parallel efficiency for solving subsurface flow in porous media with MPFA discretization, $\theta = \pi/4$.

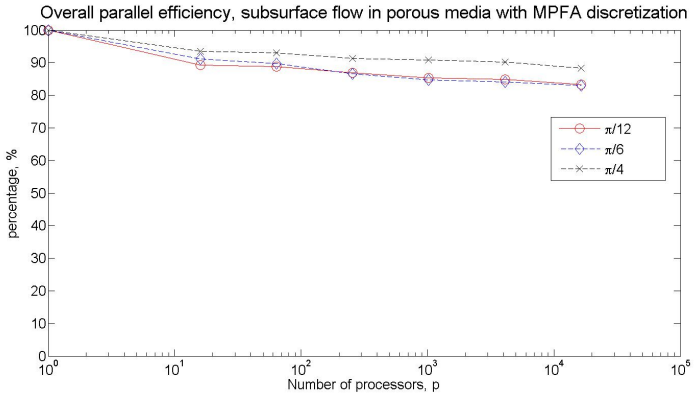


Figure 10. Overall parallel efficiency for solving subsurface flow in porous media with MPFA discretization.

On the other hand, from the experiments in **ROTANISO2D** we see severe grid size dependent convergence. The modified aggregation strategy and K-cycle help improving AGMG's performance limitedly. AGMG perform the best when $\theta = \pi/4$, in which case the aggregates can align (mostly) with the direction the anisotropy angle. For this challenging problem it is necessary to modify the aggregation strategy and the smoothers.

For problem **MPFA** we combine MPFA discretization and AGMG to solve the problem for Darcy scale subsurface flow in porous media. The grid size dependency is relatively mild and the efficiency for parallel AGMG solving is acceptable for the case $\theta = \pi/4$ ($\approx 70\%$). The parallel efficiency for MPFA discretization is excellent ($> 90\%$) in all scenarios as the algorithm (experimenting pressure field) is inherently parallel. In all the problems considered in this work, SAIPCG is a better coarsest grid solver than DPCG, although the advantage is not very significant.

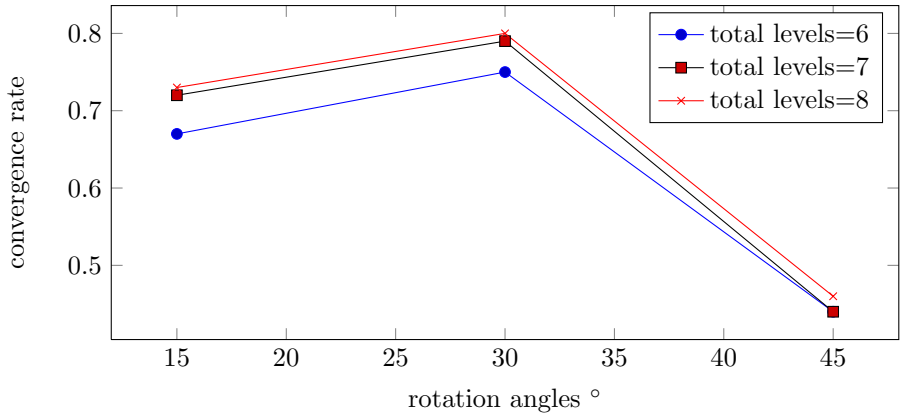


Figure 11. The relation between the average convergence rate of AGMG with the same total levels.

To the best of author's knowledge, this is the first attempt to investigate the efficiency of massively parallel AGMG for solving elliptic PDEs containing coefficients with various complexity. The current work shed light on the new approaches to tackle the complicated systems of subsurface fluid flow problem with fractures and full-tensor permeability. This work can be extended to multiphase flow in a rock containing fractures of various scales, which is a topic we plan to work on in near future.

Acknowledgments

The research reported in this publication was supported in part by funding from King Abdullah University of Science and Technology (KAUST) through the grant BAS/1/1351-01-01.

References

[1] I. Aavatsmark, T. Barkve, O. Boe and T. Mannseth, Discretization on non-orthogonal, quadrilateral grids for inhomogeneous anisotropic media. Journal of Computational Physics, 127, pp. 2-14, 1996.

[2] M. G. Edwards and C. F. Rogers, Finite volume discretization with imposed flux continuity for the general tensor pressure equation. Comput. Geosci., 2(4), pp. 259-290, 1998.

[3] I. Aavatsmark, An introduction to multipoint flux approximations for quadri-lateral grids. Comput. Geosci., 6, pp. 405 - 432, 2002.

[4] I. Aavatsmark, G. T. Eigestad, B. T. Mallison and J. M. Nordbotten, A compact multipoint flux approximation method with improved robustness. Numer. Methods for Partial Differential Equations. 24(5), pp. 1329-1360, 2008.

[5] J. Nordbotten and G. T. Eigestad, Discretization on quadrilateral grids with improved monotonicity properties. J. Comput. Phys., 203(2), pp. 744-760, 2005.

[6] S. Sun, A. Salama and M. F. El-Amin, An equation-type approach for the numerical solution of the partial differential equations governing transport phenomena in porous media. Procedia Computer Science, 9, pp. 661-669, 2012.

[7] A. Negara, A. Salama and S. Sun, 3-D Numerical Investigation of Subsurface Flow in Anisotropic Porous Media using Multipoint Flux Approximation Method. Society of Petroleum Engineering, SPE 165960, 2013.

[8] V. Reichenberger, H. Jakobs, P. Bastian and R. Helmig, A mixed-dimensional finite volume method for multiphase flow in fractured porous media. Adv. Water. Resour., 29, pp. 1030 -1036, 2006.
[9] H. Hoteit and A. Firoozabadi, An efficient model for incompressible two-phase flow in fractured media. Adv. Water. Resour, 31, pp. 891-905, 2008.
[10] M. F. El-Amin, A. Salama and S. Sun, Numerical and dimensional investigation of two-phase countercurrent imbibition in porous media. J. Comput. Appl. Math., 242, pp. 285-296, 2013.
[11] A. Zidane and A. Firoozabadi, An efficient numerical model for multicomponent compressible flow in fractured porous media. Adv. Water. Resour., 74, pp. 127-147, 2014.
[12] R. D. Falgout, P. S. Vassilevski On generalizing the algebrac multigrid framework. SIAM J. Numer. Anal., Vol. 42, No. 4, pp. 1669-1693, 2004.
[13] W. Hackbusch Multigrid Methods and Applications. Berlin: Springer, 1985.
[14] U. Trottenberg, C. Osterlee, A. Schuller, (Eds.), Multigrid, Academic Press, New York, 2000.
[15] Y. Notay, An aggregation-based algebraic multigrid method, ETNA, 37, pp. 123 - 146, 2010.
[16] A. Napov and Y. Notay, Algebraic analysis of aggregation-based multigrid, Lin. Alg. Appl., 18 (2011), pp. 539-564.
[17] Y. Notay and P. S. Vassilevski , Recursive Krylov-based multigrid cycles, Numer. Linear Algebra Appl. 2006; **0**: 1-20.
[18] Y. Notay, AGMG: Iterative solution with AGgregation-based algebraic MultiGrid, http://homepages.ulb.ac.be/ ynotay/AGMG.
[19] E. Chow, R. D. Falgout, J. J. Hu, R. S. Tuminaro and U. M. Yang A survey of parallelization techniques for multigrid solvers, Parallel Processing For Scientific Computing, SIAM, series on Software, Environments, and Tools, 2006.
[20] M. Benzi, M. Tuma, A comparative study of sparse approximate inverse preconditioners. Appl. Numer. Math., 30, pp. 305 - 340, 1999.
[21] E. Chow. Parallel Implementation and Performance Characteristics of Least Squares Sparse Approximate Inverse Preconditioners, Int. J. High-Perform. Comput. Appl., 2000.
[22] A. Napov and Y. Notay, An algebraic multigrid method with guaranteed convergence rate, SIAM J. Sci. Comput., 34(2), A1079-A1109, 2012.
[23] M. Chen, A. Greenbaum, Analysis of an aggregation-based algebraic two-grid method for a rotated anisotropic diffusion problem. Numer. Linear Alg. Appl., 22, pp. 681-701, 2015.
[24] J. Linden, G. Lonsdale, H. Ritzdorf, A. Schüller, Scalability aspects of parallel multigrid. Future Generation Computer Systems 10, pp. 429-439, 1994.
[25] Y. Notay, A massively parallel solver for discrete Poisson-like probelm, J. Comput. Phys., 281, pp. 237-250, 2015.

School of Computing, University of Leeds, Leeds, LS2 9JT, UK
E-mail: M.H.Chen@leeds.ac.uk

Physical Science and Engineering Division, King Abdullah University of Science and Technology, Thuwal 23955, KSA
E-mail: shuyu.sun@kaust.edu.sa