DOI 10.1007/s12182-014-0370-1

A multilevel preconditioner and its shared memory implementation for a new generation reservoir simulator

Wu Shuhong[1, 2*], Xu Jinchao[3], Feng Chunsheng[4], Zhang Chen-Song[5], Li Qiaoyun[2], Shu Shi[4], Wang Baohua[2], Li Xiaobo[2] and Li Hua[2]

[1] State Key Laboratory of Enhanced Oil Recovery, Beijing 100083, China
[2] Research Institute of Petroleum Exploration and Development, PetroChina, Beijing 100083, China
[3] Department of Mathematics, Penn State University, Univerisity Park, USA
[4] School of Mathematics and Computational Science, Xiangtan University, Xiangtan, Hunan 411105, China
[5] Academy of Mathematics and System Sciences, Beijing 100190 China

Abstract: As a result of the interplay between advances in computer hardware, software, and algorithm, we are now in a new era of large-scale reservoir simulation, which focuses on accurate flow description, fine reservoir characterization, efficient nonlinear/linear solvers, and parallel implementation. In this paper, we discuss a multilevel preconditioner in a new-generation simulator and its implementation on multicore computers. This preconditioner relies on the method of subspace corrections to solve large-scale linear systems arising from fully implicit methods in reservoir simulations. We investigate the parallel efficiency and robustness of the proposed method by applying it to million-cell benchmark problems.

Key words: Multilevel, preconditioner, shared memory, large-scale linear system, reservoir simulation

1 Introduction

Simulation-based scientific discovery and engineering design demand extreme computing power and highly efficient algorithms (as well as their implementations). This demand is the main driving force for developing extreme-scale computer hardware/software during the last few decades. Reservoir simulation plays an important role both in designing an efficient development process and in improving recovery factors. Significant research effort has been devoted to creating fine-scale reservoir models and efficient reservoir simulators on high-performance computers; see Hayder and Baddourah, 2012 and references therein for details.

Reservoir engineers are building high-resolution reservoir models (Pavlas, 2002; Dogru et al, 2009), on which numerous simulation runs are performed for the purpose of history matching and model validation. According to a previous case study by Saudi Aramco (Pavlas, 2002), a high-resolution model can preserve reservoir heterogeneity at a fine scale and thus maintain reservoir character and describe complex water encroachment. Saudi Aramco obtained an excellent historical validation using a 128-node cluster and the resulting predictions were in line with the company's expectations.

Despite of the increasing availability of more powerful computing resources (such as high-performance clusters), desktop computers and workstations still dominate the work environment for reservoir simulation engineers. Because of the interplay of the three "walls" —the memory wall, the instruction level parallelism wall, and the power wall (the chip's overall temperature and power consumption)— the peak performance of a single core has almost stopped improving. Even worse, single-core performance has started to deteriorate in some cases. There is a trend toward using multicore processors, which helps CPU designers to avoid the high power-consumption problem that comes with increasing chip frequency. As CPU speeds rise into the 3-4 GHz range, the amount of electrical power required is prohibitive. Hence, the trend toward multicore processors started and will continue into the foreseeable future. OpenMP is an application program interface that can be used to explicitly direct multicore (shared memory) parallelism. It is a specification for a set of compiler directives, library routines, and environment variables that can be used to specify shared memory parallelism in Fortran and C/C++ programs.

Several difficulties can arise when using multithread implementation for preconditioned Krylov subspace methods: i) Some preconditioners use sequential algorithms, like Gauss-Seidel; ii) OpenMP programs sometimes require more memory space than their corresponding sequential versions do. When a numerical algorithm is implemented in OpenMP or any other multithread computer language, it is important

*Corresponding author. email: wush@petrochina.com.cn
Received January 22, 2014

to maintain the convergence rate of the corresponding sequential algorithm. However, this is not always possible as many numerical algorithms are sequential in nature. When working with sparse matrices in compressed formats, like the Compressed Sparse Row format, we sometimes need to introduce auxiliary memory space. This becomes an increasingly heavy burden as the number of threads increases. We will analyze the parallel interpolation and coarse-grid operators in the setup phase of the algebraic multigrid (AMG) method based on the fact that the coefficient matrices we consider are banded.

In order to meet the increasing demand for high-resolution reservoir simulations in low-end desktop computing environments (multicore CPUs, sometimes with heterogeneous co-processors), we design and develop new cost-effective reservoir simulation techniques, such as different fluid models and discretization methods, for typical desktop computers. In this paper, we discuss efficient implementation of multilevel preconditioners for solving large-scale fully-implicit simulations of the black oil model. Some of the materials in this paper have been previously presented in two conference proceedings by the authors (Wu et al, 2013a; Feng et al, 2014) and we repeat them for completeness. The main contribution of this paper is the new numerical study on the OpenMP performance of the multilevel FIM solver, which has not been seen in the literature.

The rest of this paper is organized as follows: In Sec. 2, we briefly review the mathematical model and its fully implicit discretization method. We discuss the data structure for block sparse matrices in Sec. 3, after which we introduce a preconditioner based on a successive subspace correction framework in Sec. 4. We discuss several implementation issues of the proposed algorithm using OpenMP. We then perform a numerical experiment to test efficiency, robustness, and multicore speed-up of the proposed preconditioner in Sec. 6. Finally, we summarize the discussion with a few concluding remarks in Sec. 7.

2 Mathematical model and its fully implicit discretization

Most of China's oil fields are located in continental basins and many are characterized by serious heterogeneity, low permeability, and high oil viscosity (Han, 1998; Han et al, 1999). Water breakthrough occurs at an early stage of development in these fields, which results in low recovery efficiency even when enhanced water injection techniques are employed. Higher resolution reservoir modelling is needed to analyze complex flow phenomena in these fields.

The black oil model is often applied in the primary and secondary oil-recovery stages. In this model, the fluid is assumed to have three quasi-components (Water, Oil, and Gas) and they form three respective phases (water, oil, and gas): The water phase does not exchange mass with the other phases, and the liquid and gaseous phases exchange mass with each other. As a widely accepted approach, the isothermal black oil model solves the three-dimensional three-phase equations of the conservation of mass (volume) in porous media in the standard surface conditions, subject to appropriate initial and boundary conditions (Chen et al, 2006). The material balance of the hydrocarbon gaseous (gas), liquid (oil), and water components is described, respectively, by

$$\frac{\partial}{\partial t}\left[\phi\left(\frac{S_{\mathrm{g}}}{b_{\mathrm{g}}}+\frac{R_{\mathrm{s}}S_{\mathrm{o}}}{b_{\mathrm{o}}}\right)\right]+\nabla\cdot\left(\frac{1}{b_{\mathrm{g}}}u_{\mathrm{g}}+\frac{R_{\mathrm{s}}}{b_{\mathrm{o}}}u_{\mathrm{o}}\right)=q_{\mathrm{G}} \tag{1}$$

$$\frac{\partial}{\partial t}\left(\phi\frac{S_{\mathrm{o}}}{b_{\mathrm{o}}}\right)+\nabla\cdot\left(\frac{1}{b_{\mathrm{o}}}u_{\mathrm{o}}\right)=q_{\mathrm{O}} \tag{2}$$

$$\frac{\partial}{\partial t}\left(\phi\frac{S_{\mathrm{w}}}{b_{\mathrm{w}}}\right)+\nabla\cdot\left(\frac{1}{b_{\mathrm{w}}}u_{\mathrm{w}}\right)=q_{\mathrm{W}} \tag{3}$$

It is assumed that the linear Darcy law governs the fluid flow of each phase in porous media:

$$u_{\alpha}=-\frac{kk_{\mathrm{r}\alpha}}{\mu_{\alpha}}\left(\nabla P_{\alpha}-\rho_{\alpha}g\nabla z\right),\quad \alpha=\mathrm{o},\mathrm{g},\mathrm{w} \tag{4}$$

The phase saturations appear to satisfy the condition

$$S_{\mathrm{o}}+S_{\mathrm{g}}+S_{\mathrm{w}}=1 \tag{5}$$

We further assume that the capillary pressures characterize the pressure differences between phases:

$$P_{\mathrm{o}}-P_{\mathrm{w}}=P_{\mathrm{cow}},\ \ P_{\mathrm{g}}-P_{\mathrm{o}}=P_{\mathrm{cgo}} \tag{6}$$

Remark 1 In this paper, we focus on the fully implicit method for solving the above black oil model. The methods discussed here can be readily extended to other models and numerical discretization schemes. For example, a new-generation simulator can also handle the well-known volatile oil model in which the oil mass balance equation (Eq. (2)) is replaced by

$$\frac{\partial}{\partial t}\left[\phi\left(\frac{S_{\mathrm{o}}}{b_{\mathrm{o}}}+\frac{R_{\mathrm{v}}S_{\mathrm{g}}}{b_{\mathrm{g}}}\right)\right]+\nabla\cdot\left(\frac{1}{b_{\mathrm{o}}}u_{\mathrm{o}}+\frac{R_{\mathrm{v}}}{b_{\mathrm{g}}}u_{\mathrm{g}}\right)=q_{\mathrm{O}} \tag{7}$$

Before we start to discuss how to solve the above equations, we first make a few comments. The phase injection rate and the total liquid production rate constraints can be applied for the wells. When the bottom-hole pressure P_{bh} cannot sustain the fixed flow rate, the well equation automatically switches to the fixed bottom-hole pressure case. For simplicity, we will not present details pertaining to the treatment of well constraints. When the reservoir pressure drops below the bubble-point pressure (undersaturated state), the hydrocarbon phase splits into a liquid (oil) phase and a gaseous (gas) phase at the thermodynamical equilibrium. In this case, we choose P_{o}, S_{w}, and S_{g} as the primary variables, with the rest of the unknowns represented by the primary variables using Eqs. (5) and (6). On the other hand, if the gas phase is not present (saturated state), we use R_{s} instead of S_{g} as a primary variable. However, we will not consider the

saturated case when presenting the algorithm in this paper.

Among many possible methods for the above model, we will consider only the fully implicit method (FIM). The fully implicit method (Douglas et al, 1959) is a discretization method that is often used to solve the black oil model in petroleum reservoir simulators. In FIM, Newton linearization is combined with first-order upstream-weighting finite difference spatial discretization (for details, see Chapter 8 in Chen et al, 2006). This scheme is accurate and stable, as proved by several decades of practical usage. The main disadvantage of FIM is the computational cost associated with solving the Jacobian systems arising from Newton's method. Very often, solving such linear systems with direct or iterative solvers takes more than 80% of the computational time in reservoir simulation. Furthermore, the demand for more accurate computer simulation has led to larger and, in turn, more heterogeneous field-scale reservoir models. Such models entail larger and more difficult linear systems.

3 Storage format for block sparse matrices

The Jacobian systems arising from the Newton linearization in FIM are usually large, sparse, nonsymmetric, and ill conditioned. Krylov subspace methods (Saad, 2003), such as BiCGstab and GMRes, are efficient iterative methods for solving these Jacobian systems. Many preconditioning techniques have been proposed for reservoir simulation (see, for example, Dupont et al, 1968; Meijerink and van der Vorst, 1977; Appleyard et al, 1981; Behie and Vinsome, 1982; Appleyard and Cheshire, 1983; Meijerink, 1983; Wallis, 1983; Behie and Forsyth, 1984; Concus et al, 1985; Wallis et al, 1985; Lacroix et al, 2001; 2003; Watts and Shaw, 2005; Stüben et al, 2007; Al-Shaalan et al, 2009; Hu et al, 2013b; Wang et al, 2013a; 2013b).

When FIM is combined with the cell-center finite difference method, a fully coupled linear algebraic system

$$Au = f, \quad \text{i.e.} \begin{pmatrix} A_{\text{ResRes}} & A_{\text{ResWel}} \\ A_{\text{WelRes}} & A_{\text{WelWel}} \end{pmatrix} \begin{pmatrix} u_{\text{Res}} \\ u_{\text{Wel}} \end{pmatrix} = \begin{pmatrix} f_{\text{Res}} \\ f_{\text{Wel}} \end{pmatrix} \tag{8}$$

must be solved in each Newton step. Here, the subscripts 'Res' and 'Wel' stand for the reservoir and implicit well parts, respectively, of the main solution variables. Let m be the number of unknowns in each grid cell. For example, in FIM for the black oil model, m is equal to 3; and m is equal to 2 for the dead oil case (no gas phase). Assume that there are N active grid cells and M implicit wells. Then the size of the Jacobian matrix is $mN+M$. Specifically, the solution vector space is $V=R^{mN+M}$.

Remark 2 Because the well part and the main reservoir part differ in regard to shape, the Jacobian matrix A is sometimes referred to as the bordered matrix. In practice, we have found that many iterative solvers converge slowly or even fail to converge for practical problems. The coupling between the reservoir equations and the well constraints is usually strong. Based on this observation, we extend all the implicit well blocks such that they have the same dimension as the reservoir block by introducing artificial auxiliary saturation variables to each implicit well block. At the same time, we pad the right-hand side with zeros at the corresponding positions of these artificial variables to obtain $\bar{f} \in \bar{V} = \boldsymbol{R}^{m(N+M)}$. We then group the oil pressure together with the well bottom-hole pressure together and further write the local Jacobian matrix (for one grid cell) in the following form:

$$A_{ij} = J = \begin{pmatrix} J_{PP} & J_{PS} \\ J_{SP} & J_{SS} \end{pmatrix} \in \boldsymbol{R}^{m\times m} \tag{9}$$

where P denotes the pressure variables (oil pressure and the well bottom-hole pressure) and S denotes the saturation variables (including physical water and oil saturations for the reservoir blocks and artificial saturations for the implicit well blocks).

This way we can store the expanded coefficient matrix $\bar{A} = (A_{ij}) \in \boldsymbol{R}^{n\times n}$ with $n = m(N+M)$ in a uniform BSR format. To store the coefficient matrix in a cost-effective way, we employ a block sparse matrix data structure, which is often used for numerically simulating PDE systems, i.e., the block compressed sparse row (BSR) data structure. The BSR format is a generalization of the well-known compressed sparse row (CSR) format and is used in many numerical software packages, including the Intel MKL sparse direct solver library and the NIST sparse BLAS library. The difference between BSR and CSR is that in BSR, each nonzero entry is an array real numbers of size m^2, instead of one real number as in CSR. This array represents the small, dense Jacobian matrix in each grid cell. Of the several variants of the BSR format, the following triple-array definition is used in the present paper as well as in a new-generation simulator:

- val: A real array that contains the elements of the non-zero blocks of a sparse matrix. The elements are stored block by block in row-major order. All the elements of the non-zero blocks are stored, even the elements that are equal to zero. Within each non-zero block, the elements are stored in row major order.
- col: Entry i of this integer array is the number of the column in the block matrix that contains the i-th non-zero block.
- row: Entry j of this integer array is the index of the entry in col that is the first non-zero block in the j-th row of the block matrix.

As a pictorial example, we show a simple 3×3 grid block system (Fig. 1) with a vertical well located at (2, 5, 8) with 2 and 5 perforated. After expansion, we obtain a block sparse matrix where each nonzero block is a 3×3 matrix for the black oil model. We then store this matrix in the BSR format described above. Note that this modification will not introduce much extra storage or computational cost as the number of implicit wells is usually small compared to the size of the reservoir system.

An important reason why we choose the BSR format for our implementation is that it can improve the parallel scalability for sparse matrix-vector multiplications (SpMV), which is the most time-consuming part in iterative linear solvers and it usually takes most of the CPU time. A lot of research has been devoted to improve SpMV; see Oliker et al, 2002 and references therein for related discussions.

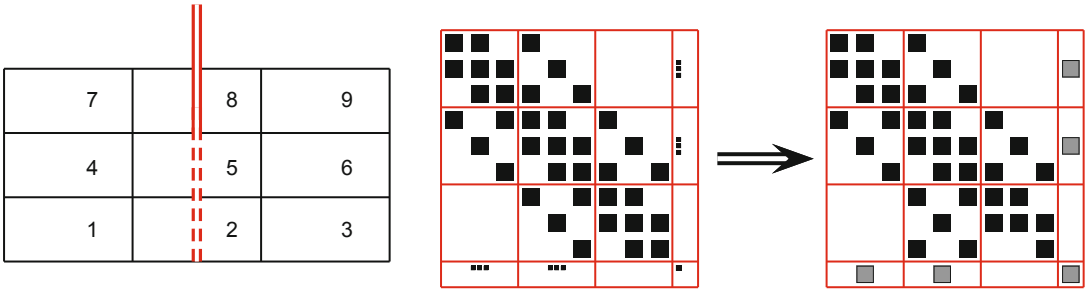

Fig. 1 Left: A 2D 3-by-3 grid with a vertical well at the center; Right: Expansion of Jacobian matrix (the last row and column for well constraint are expanded to the BSR format)

Usually different computer architectures require different SpMV implementation in order to maximize performance. Here we focus on the sparse Jacobian matrices from the fully implicit reservoir simulation. For general purpose SpMV implementations, interested readers are referred to Lam et al, 1991; Bjørstad et al, 1993; Vuduc, 2003; Wang et al 2013a. In Table 1, we use Jacobian matrices, arising from a three-phase black oil simulation on a mesh with 3.2 million active celles (about 9.6 million degrees of freedom). In the table, "Ratio CSR/BSR" means the ratio between wall times taken by 100 times of CSR and BSR sparse matrix-vector multiplication operaions. From this table, we immediately see the advantages of the BSR format over the CSR format. The BSR SpMV not only takes less computation time, but also yields better parallel speed-up.

Table 1 SpMV (100 times) using the CSR and BSR sparse matrix formats

Number of OpenMP	CSR format		BSR format		Ratio
threads N_T	Wall time, s	Speed-up	Wall time, s	Speed-up	CSR/BSR
1	29.52	1.00	26.40	1.00	1.12
2	17.43	1.69	13.17	2.00	1.32
4	10.68	2.76	8.77	3.01	1.22
8	8.21	3.60	6.61	3.99	1.24

4 A successive subspace correction preconditioner for FIM

It is self-evident that different parts of A have different algebraic properties (Trangenstein and Bell, 1989). The part corresponding to the pressure unknowns is elliptic and the part corresponding to the saturation unknowns is mainly hyperbolic. Based on this understanding, CPR-type preconditioners (Wallis, 1983; Wallis et al, 1985) take advantage of this property and become a competitive alternative in reservoir simulation. A subspace space method (MSC) has been proposed and discussed by Hu et al (2013b) where each auxiliary space solver takes these algebraic properties into account. In this paper, we focus on the multi-thread implementation of this preconditioner for solving large-scale linear systems arising from the fully implicit reservoir simulations analyzed by Hu et al (2013b). We now briefly review this preconditioner for completeness.

Remark 3 As suggested by many researchers (Bank et al, 1989; Lacroix et al, 2001; Stüben et al, 2007; Al-Shaalan et al, 2009), in order to weaken the strong coupling between the pressure and saturation unknowns, a decoupling step has to be applied to Eq. (8). This decoupling procedure should be computationally cheap and weaken the coupling between the pressure and saturation unknowns. Here, we choose the so-called alternative block factorization (ABF) strategy (Bank et al, 1989). This strategy is basically block diagonal preconditioning: $\tilde{A} = D^{-1}\overline{A}$ and $\tilde{f} = D^{-1}\overline{f}$. Here, D stands for the block diagonal matrix of the expanded matrix $\overline{A}$, i.e., $D = (A_{ii}) \in \boldsymbol{R}^{n\times n}$. In the rest of the paper, we will use the notation and write the new matrix $\tilde{A}$ as $A \in \boldsymbol{R}^{n\times n}$ with n=m(N+M). The same convention also applies to the right-hand side vector $f \in \boldsymbol{R}^n$ and to the solution $u \in \boldsymbol{R}^n$.

For the black oil model, we consider two subspace spaces: $\boldsymbol{V}_P \subset \boldsymbol{V}$ and $\boldsymbol{V}_S \subset \boldsymbol{V}$. Here, $\boldsymbol{V}_P$ is the vector space for the pressure variables (including P_o for the oil phase and the bottom-hole pressure for the implicit wells) and V_S is the vector space for the saturation variables (S_w and S_g, respectively, for the reservoir grid cells and artificial variables for the implicit wells). We have the following multiplicative version of the MSC algorithm:

Algorithm 1 (MSC Preconditioner) Given a vector u, we define a preconditioning action Bu as follows:

1) $u_0 = u$

2) $u_1 = u_0 + \Pi_S B_S \Pi_S^* (f - Au_0)$

3) $u_2 = u_1 + \Pi_P B_P \Pi_P^* (f - Au_1)$

4) $u_3 = u_2 + R(f - Au_2)$

5) $Bu = u_3$

Here $\Pi_P : \boldsymbol{V}_P \to \boldsymbol{V}$ and $\Pi_S : \boldsymbol{V}_S \to \boldsymbol{V}$ are the inclusion operators and the superscript $*$ denotes the adjoint operator, which is simply the transpose operator if applied to matrices. For example, $\Pi_P^* : \boldsymbol{V} \to \boldsymbol{V}_P$ is the injection operator from the whole space to the pressure variable space. Note that we only apply the operator B as a preconditioner and that there is no reason to solve the sub-problems exactly. Therefore, in practice, we replace A_{PP}^{-1} and A_{SS}^{-1} by the preconditioners (or simple iterative methods) B_P and B_S, respectively. In

Step 4), we introduce a smoother R for the original solution space. Note that different subspace solvers yield different preconditioners. We should choose appropriate subspace solvers according to the characteristic of the problem and the computer hardware.

The saturation variables $S=(S_w, S_g)$ have hyperbolic characteristics. Due to this fact, we solve the saturation block by the block Gauss-Seidel method. To improve the convergence rate, one can apply the Gauss-Seidel method with downwind ordering and crosswind blocks (see Wang and Xu, 1999 for details). This method orders the gridblocks according to the direction of the multiphase flow and it has been shown to be efficient for convection-dominated problems. However, in order to obtain better parallel speed-up, we use a simple block Gauss-Seidel method with multi-color ordering (see Feng, 2014 for details). Note that this choice is for better parallel scalability instead of improving convergence rate. From this we can see the infrenence of computer architecture on the choice of numerical algorithms.

It is well-known that the equations describing the mass balance in terms of pressure unknowns P are mainly elliptic (Wallis et al, 1985; Lacrois et al, 2003; Stüben et al, 2007; Al-Shaalan et al, 2009; Hu et al, 2011). Therefore, we use the algebraic multigrid (AMG) methods (Brandt et al, 1985; Ruge and Stüben, 1987; Stüben, 2001; Falgout, 2006) to solve the pressure block A_{PP}. In this paper, we use the classical AMG method for simplicity. In practice, the performance and efficiency of AMG may degenerate when the physical and geometric properties of the problems become more complicated. In order to improve the performance of the AMG solver, Hu et al (2013a) have developed an approach that combines an iterative method with some other preconditioner to obtain a new solver for the pressure block.

The smoother R in the algorithm resolves the coupling between the pressure unknowns and the saturation unknowns, as well as the coupling between the reservoir unknowns and well unknowns. The line successive over-relaxation (LSOR) method and the block incomplete factorization (BILU) methods have been applied in reservoir simulations and are often used in practice. The convergence rate of both LSOR and BILU are noticed to deteriorate when the size of the problems increases, or when the porous media become more heterogeneous. LSOR requires geometric information from the underlying mesh. BILU(k) with a large fill-in level k may become too expensive (in terms of memory usage) in practice for large-scale simulations. Therefore, to reduce computational cost (both CPU time cost and memory cost), the block Gauss-Seidel method is used as the smoother in this paper.

5 OpenMP implementation and shared memory paradigm

In this section, we discuss OpenMP parallel implementation of the proposed preconditioner in Algorithm 1 on typical desktop computers with multicore CPU's. Compared to message-passing implementations like MPI, the shared memory paradigm can greatly simplify the programming task in a multicore environment. OpenMP parallel programs are relatively easy to implement, as each processor has a global view of the entire memory. Parallelism can be achieved by inserting standard compiler directives into the code to distribute loop iterations among the processors. However, performance may suffer from poor spatial locality of physically distributed shared data. We now focus to the setup stage of the Classical AMG method. Notice that the AMG method is applied to the pressure equation only and we use the standard CSR sparse matrix format for the coefficient matrices. Feng et al (2014) proposed a simple but efficient algorithm for constructing standard prolongation and coarse-level operators using OpenMP. If the bandwidth of the sparse coefficient matrix A is relatively small, this algorithm can save a large amount of memory.

To simplify the notation, we denote the coefficient matrix A_{PP} as $A \in \boldsymbol{R}^{n\times n}$ in this section. Let $G_A(V, E)$ be the graph of A, where V is the set of vertices (i.e., unknowns) and E is the set of edges (i.e., connections that correspond to nonzero off-diagonal entries of A). Assume that the index set of vertices is split into two sets: a set C of coarse-level vertices and a set F of fine-level vertices, such that

$$V = C \bigcup F \quad \text{and} \quad C \bigcap F = \varnothing$$

We denote n_c as the cardinality of C, i.e., the number of C-vertices. Assume that F^c is the map from F-vertices to C-vertices. We define the set of neighboring variables of i as $N_i := \{j \in V : A_{ij} \neq 0, j \neq i\}$.

For a fixed real number $\theta \in [0, 1)$, we denote the strong-connected variables as

$$S_i(\theta) := \left\{ j \in N_i : -A_{ij} \geq \theta \max_{k\neq i}(-A_{ik}) \right\} \tag{10}$$

Let $D_i^{F,s} := S_i(\theta) \cap F$, $D_i^{C,s} := S_i(\theta) \cap C$, $D_i^{w} := N_i \setminus (D_i^{C,s} \cup D_i^{F,s})$. We can now define

$$F_i := \left\{ j \in D_i^{F,s} : i \text{ and } j \text{ without the same depended } C\text{-vertices} \right\}$$

Let $\hat{A}_{ij} := 0$ if $A_{ii}A_{ij} > 0$, and $\hat{A}_{ij} := A_{ij}$, otherwise. We denote the standard prolongation (or interpolation) matrix as $P = (P_{ij_c}) \in \boldsymbol{R}^{n\times n_c}$, where its entries can be determined as follows:

$$P_{ij_c} = \begin{cases} -\left(A_{ij} + \sum\limits_{k\in D_i^{F,s}\setminus F_i} \dfrac{A_{ik}\hat{A}_{kj}}{\sum\limits_{m\in D_i^{C,s}} \hat{A}_{km}}\right) \Big/ \left(A_{ii} + \sum\limits_{k\in D_i^{w}\cup F_i} A_{ik}\right) \\ \qquad\qquad i \in F,\ j \in D_i^{C,s},\ j_c = F^C[j] \\ 1.0, \quad i \in C,\ j_c = F^C[i] \\ 0.0, \quad \text{otherwise.} \end{cases}$$

The matrix P is sparse and is usually stored in the CSR format, we need an auxiliary integer marker called M_p to locate the column index of each non-zero entry. To generate the i-th row of P, we define, for $0 \leq j \leq n-1$, that

$$M_P[j] := \begin{cases} J_{j_c},\ j \in D_i^{C,s}, \quad j_c = F^C[j] \\ -2-i, \quad j \in D_i^{F,s} \setminus F_i \\ -1, \quad \text{otherwise} \end{cases} \tag{11}$$

where J_{j_C} is the position of P_{ij_C} entry in the column index array of the CSR storage of P. In the OpenMP implementation, we have to allocate an integer array for the marker M_P for each OpenMP thread. The length of each M_P is n , and the total length of M_P of all threads is then $N_T \times n$ where N_T is the total number of threads.

Assume that $b_n = b_l + b_r$ is the bandwidth of A, where b_l and b_r are the left- and right-bandwidth of the coefficient matrix A, respectively. When the parallel partition of V is continuously distributed in a balanced fashion to each OpenMP thread (i.e., the size difference on each thread does not exceed one), we can easily see that the number of entries of M_P that are actually used by the program is much smaller than n (see Fig. 2 for an example). Taking into account the fact that the matrix is banded, we can get the following estimates of the length L_P^t and the minimal offset $M_l^t(P)$ (Feng et al, 2014)

$$L_P^t \le \min(n, \frac{n}{N_T} + 2b_n) \quad \text{and} \quad M_l^t(P) \ge \max\left(0, \frac{n}{N_T}(t-1) - 2b_n\right) \tag{12}$$

The coarse grid operator of multigrid methods can be built using the Galerkin relation $A_c = (A_{ij}^c)_{n_c \times n_c} := P^{\mathrm{T}} AP$, where

$$A_{ij}^c = \sum_{k_1} \sum_{l_1} P_{k_1 i} A_{k_1 l_1} P_{l_1 j}, \qquad i, j = 1, \mathrm{L}\ , n_c. \tag{13}$$

Similar to the implementation of the prolongation operator, we need to allocate two auxiliary integer arrays called M_A and M_P (see Fig. 3 for a pictorial demonstration). The length of M_A and M_P are n and n_c, respectively. By noticing the characteristic of the banded sparse matrices of the coarse operator, we can get the estimation formula for the lengths of M_A and M_P. The actual needed length L_A^t and the offset $M_l^t(A)$ can be calculated using the following formulas

$$L_A^t \le \min(n, \frac{n}{N_T} + 2b_n) \quad \text{and} \quad M_l^t(A) \ge \frac{n}{N_T} t - b_n \tag{14}$$

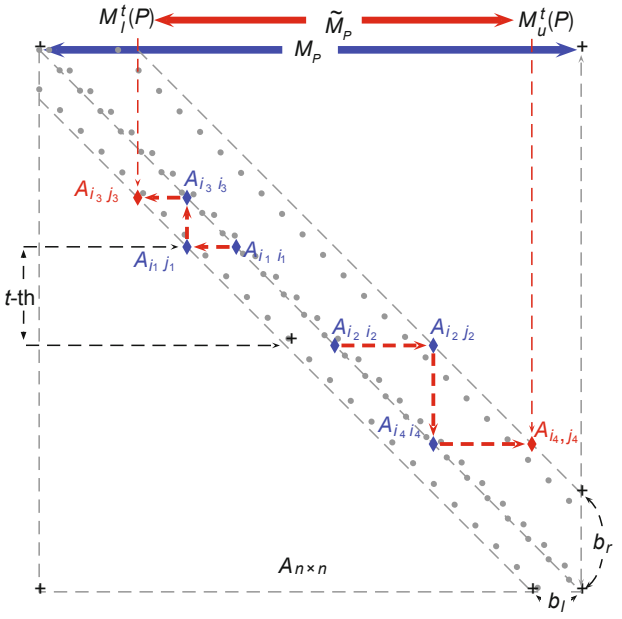

Fig. 2 Construction of the prolongation for A. $M_l^t(P)$ and $M_u^t(P)$ are the lower and upper, respectively, column indices of the non-zero entries of A of the t-th OpenMP thread.

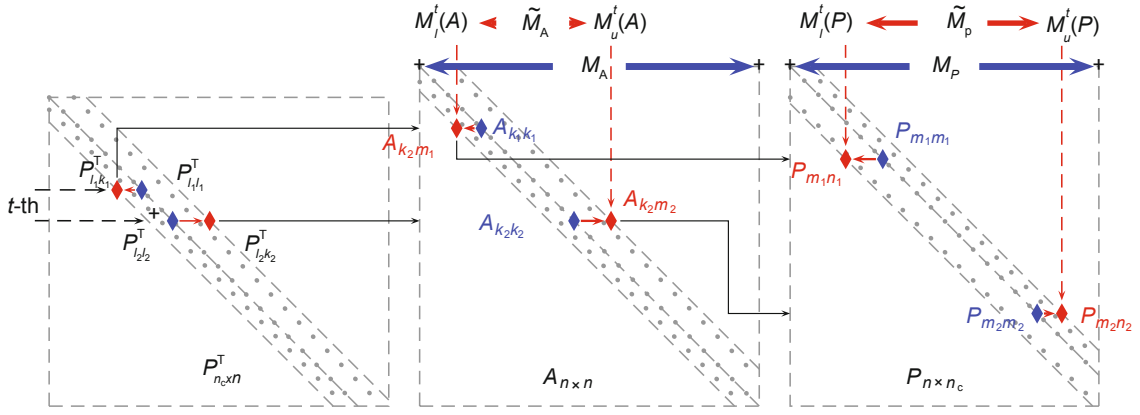

Fig. 3 Construction of the Galerkin coarse-level operator $A_c = P^{\mathrm{T}} AP$. $M_l^t(A)$ and $M_u^t(A)$ are the lower and upper column indices of the non-zero entries in A of the t-th OpenMP thread.

6 Numerical expriments

We use the second model from the Tenth SPE Comparative Solution Project (Christie and Blunt, 2001), which was designed to compare the ability of upscaling approaches used by various participants to predict the performance of water flooding in a highly heterogeneous black-oil reservoir with simple geometry described by a fine-scale (60×220×85=1,122,000) regular Cartesian geological model. The model described herein was originally generated for use in the PUNQ project. The problem statement specified that the competition's purpose was to compare the respective solutions in regard to accuracy. The model dimensions are 1200×2200×170 (ft). There is one injector at the center of the field and four producers, one at each of the four corners. The total simulation time is 2,000 days.

The model has no top structure or faults and has a uniform initial water-oil interface. The depth of the reservoir is 3,657.6 m, and the initial field pressure is 41.37 MPa. Oil density at the standard condition is 0.849 g/cm^3 and oil viscosity at the reservoir condition is 3 mPa·s. The field has a low saturation pressure, and there are only two phases (water and oil) during the whole simulation. The top 21.35 m (35 layers) represents the Tarbert formation, and the bottom 30.5 m (50 layers) represents the Upper Ness. The top part of the model

represents a prograding near-shore environment and the lower part is fluvial. In this model problem, the permeability range is 0.66×10^{-3}-20000×10^{-3} μm^2 and the average permeability is 364.52×10^{-3} μm^2. The ratio between the vertical and horizontal permeabilities, k_v/k_h, varies from 0.001 to 0.3. The average and maximal porosity values of the field are 0.1749 and 0.5, respectively. See Fig. 4 for the porosity of each of the four sample horizontal layers.

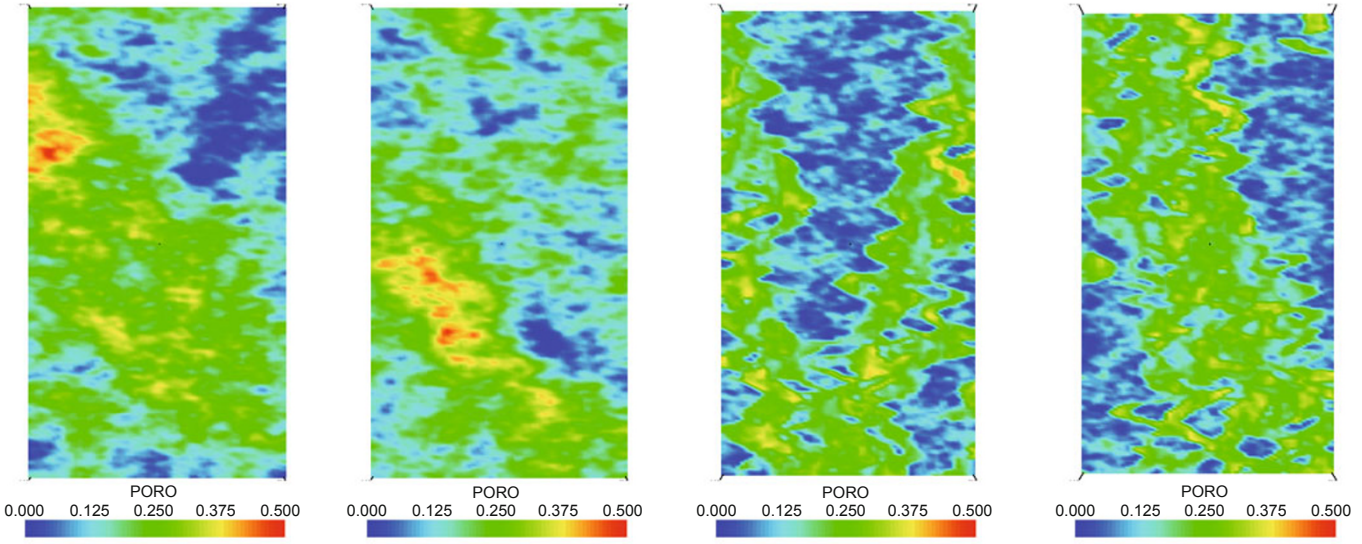

Fig. 4 Porosity of four sample horizontal layers in SPE10 (Model 2)

The simulation was performed first on a Dell desktop PC with Intel Core i7 3.33 GHz CPU (4 cores) and 8 GB DDR3 RAM. This test platform (Platform A) cost about $1,250 USD when bought new in early 2011. The Intel Core i7 utilizes the hypre-threading (HT) technology, which was developed to improve parallel performance by duplicating certain sections of the processor. However, some experiments have indicated that HT might cause loss of efficiency for some applications (Abdel-Qader and Walker, 2010). In our experiments, we disable the HT feature of the i7 CPU. It is well-known that the parallel efficiency heavily depends on algorithm, implementation, and hardware architecture. We use another computer (Platform B) for comparison: HP Z800 server with two Intel Xeon X5590 CPU (4 cores) and 24 GB DDR3 RAM. This computer was purchased early in 2010 and the market price at that time was $7,000 USD. A single core of Intel Xeon X5590 is much less powerful than the Intel i7 CPU.

The total wall time for a single simulation run using a new-generation simulator, HiSim, is less than 45 minutes for the SPE10 problem (1.1 M grid cells, 2.2 M degrees of freedom) using one single thread (detailed numerical results will be reported in Table 2 and further discussed later). HiSim is an in-house reservoir simulator, developed by RIPED, PetroChina, with several fluid flow models and the multilevel preconditioner discussed in this paper implemented therein; see, Li et al, 2013a; 2013b; Wu et al, 2013a; 2013b. And, the linear solver alone takes about 85% of the total simulation time of a new-generation simulator when using one core only. We compared our numerical results with the benchmark results by Landmark, Geoquest, Chevron, and Streamsim reported in Christie and Blunt, 2001 (Figs. 5-6). The curves of field oil rate, field average pressure, well oil rate, and well water-cut in the figures show good agreement with the reported results using other simulators.

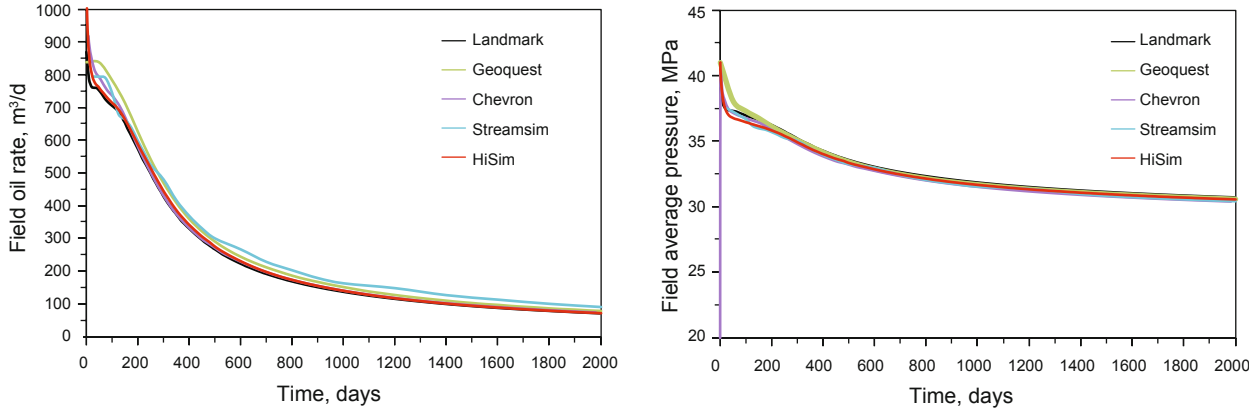

Fig. 5 Comparison of field oil rate (Left) and average pressure (Right) by different simulators

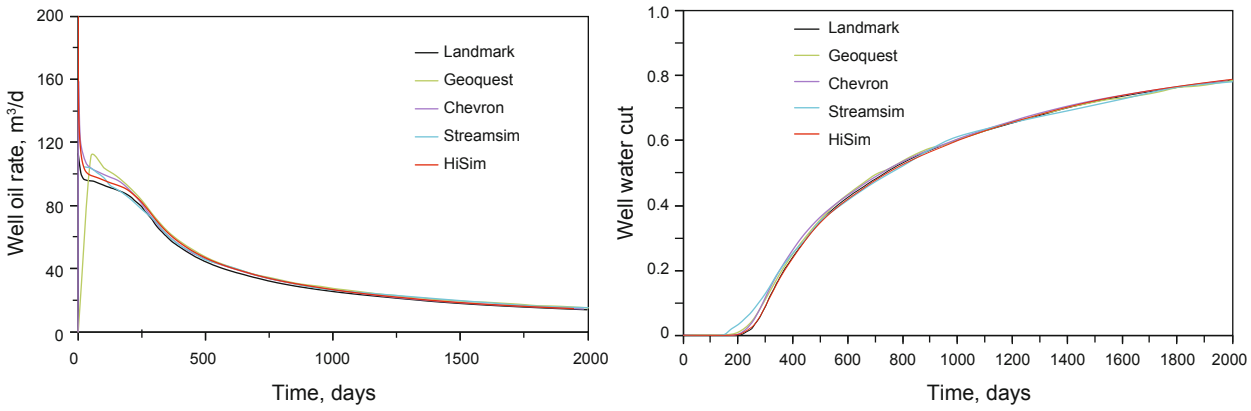

Fig. 6 Comparison of oil rate (Left) and water cut (Right) in Producer 1 by different simulators

The large problem size and heterogenous nature of the benchmark make it very challenging; as a result, it is suitable for testing algorithm efficiency, robustness, and parallel speed-up of the proposed preconditioner. As we mentioned earlier, Algorithm 1 results in various preconditioning strategies by choosing different subspace solvers or smoothers. In this section, we only compare the performance of three simple choices: the original preconditioner B in Algorithm 1, a simplified version B_1 by neglecting Step 4 of Algorithm 1 (i.e., without the global smoothing step or R =0), and another simplified version B_2 by neglecting Step 2 of Algorithm 1 (i.e. B_S=0). We note that B_2 is in fact the CPR method if the global smoother R is chosen to be the ILU or Block ILU method.

We set the stopping criteria to be the relative residual in the Euclidian norm less than 10^{-3}. In Table 2, we summarize the performance of a new-generation simulator, in which #Timesteps is the total number of time steps, #Newton is the total number of Newton iterations, #Linear is the total number of linear iterations, wall time is the total wall-time for the whole simulation (including I/O operations), Average #Newton is the average number of Newton iterations in each time step, and Average #Linear is the average number of linear iterations in each Newton iteration.

Table 2 Comparison of the preconditioned GMRes methods for SPE10

Preconditioner	#Timesteps	#Newton	#Linear	Average #Newton	Average #Linear	Wall time, min
B	161	254	2508	1.58	9.87	41.58
B_1	161	286	3773	1.78	13.19	53.50
B_2	161	269	4462	1.67	16.59	59.19

From Table 2, we find that each component of the preconditioner B plays a role in the convergence of the iterative method. Removing the smoother B_S or R will not only cause the average number of linear iterations (#Linear) to increase, but also cause the total number of nonlinear iterations (#Newton) to increase slightly. Although our choice for the components of the proposed algorithm might not yield the best preconditioner for all problems, it is quite efficient and robust for this challenging benchmark problem.

Next we investigate the OpenMP speed-up of the preconditioned GMRes method discussed in Sec. 5. The implementation is done by adding OpenMP directives to our simulator code and it has only been done for the most time-consuming part of the linear solver. The numerical results (total number of Newton steps, average number of linear iterations, total wall time in minutes, and parallel speed-up) are reported in Table 3. The parallel speed-up (the ratio between the simulation time using one core over the simulation time on multiple cores) is 1.37 when using 4 threads on the 4-core i7 CPU. And speed-up of the solver part is about 1.5 folds. Note that the solver part is the only place where OpenMP directives are employed.

Table 3 OpenMP performance of the preconditioned GMRes solver for SPE10 on Platform A

Number of OpenMP threads N_T	Total Newton steps	Average linear iterations	Wall time min	Linear solver time min	Parallel speed-up (Linear solver)
1	254	9.87	41.58	35.36	1.00
2	262	10.19	32.60	25.61	1.38
4	260	10.00	30.45	23.23	1.52

We can expect that, when using one thread, the simulation on Platform B will take more CPU time than on Platform A. The numerical results confirm this expectation; see Tables 3 and 4. However, we also notice that Platform B has much better parallel efficiency and the solver parallel speed-up is substantially improved; see Table 4. This example shows the importance for users to take full advantage of modern computers by explore parallelism in their algorithms and implementations.

Table 4 OpenMP performance of the preconditioned GMRes solver for SPE10 on Platform B

Number of OpenMP threads N_T	Total Newton steps	Average linear iterations	Wall time min	Linear solver time min	Parallel speed-up (Linear solver)
1	254	9.87	50.08	46.13	1.00
2	262	10.19	41.25	30.28	1.52
4	260	10.00	32.78	20.82	2.22
8	261	10.67	32.40	20.42	2.26

7 Summary and conclusions

We discussed a practical and efficient preconditioner for large sparse linear systems arising from the black oil model discretized by the fully implicit method. The method of subspace corrections was used to construct a new preconditioner of the original highly coupled Jacobian system by several sub-problems and suitable solution techniques to approximate these sub-problems according to their analytic characteristics. The new method can be used as a preconditioner for Krylov subspace iterative methods. The results of the preliminary numerical experiments show that the linear algebraic solver is quite efficient and robust for highly heterogeneous benchmark and field-scale problems. This new solution technique can achieve a turnaround time on a new-generation simulator for a million-cell model of less than an hour on a mainstream desktop computer. The performance of the solution method on a shared memory multicore environment is reasonably good for relatively large reservoir simulation models. Further code optimization is required to improve parallel efficiency.

Acknowledgements

The authors would like to thank RIPED, PetroChina, for providing data for the numerical tests and support through PetroChina New-generation Reservoir Simulation Software (2011A-1010), the Program of Research on Continental Sedimentary Oil Reservoir Simulation (z121100004912001) founded by Beijing Municipal Science & Technology Commission and PetroChina Joint Research Funding12HT1050002654. Feng is partially supported by the NSFC Grant 11201398, and Hunan Provincial Natural Science Foundation of China Grant 14JJ2063 and Specialized Research Fund for the Doctoral Program of Higher Education of China Grant 20124301110003. Zhang is partially supported by the Dean's Startup Fund, Academy of Mathematics and System Sciences and the State High Tech Development Plan of China (863 Program) 2012AA01A309. Shu is partially supported by NSFC Grant 91130002 and Program for Changjiang Scholars and Innovative Research Team in University of China Grant IRT1179 and by the Scientific Research Fund of the Hunan Provincial Education Department of China Grant 12A138.

Nomenclature

α	oil (o), gas (g) and water (w) phases
β	Oil (O), Gas (G) and Water (W) components
P_α	pressure of α phase, MPa
S_α	saturation, fraction
u_α	velocity, m/s
ϕ	porosity, fraction
k	absolute permeability, $10^{-3}\mu m^2$
$k_{r\alpha}$	relative permeability, fraction
μ_α	viscosity, mPa·s
b_α	formation value factor, m^3/m^3
ρ_α	fluid density, kg/m^3
q_β	source/sink term (wells), m^3/d
P_{cow}, P_{cgo}	capillary pressures, MPa
R_s	solution gas-oil ratio, m^3/m^3
R_v	oil volatility, m^3/m^3
g	gravitational acceleration, m/s^2
z	depth,m

References

Abdel-Qader J H and Walker R S. Performance evaluation of OpenMP benchmarks on Intel's quad core processors, Proceedings of the 14th WSEAS International Conference on Computers, 348-355, 2010

Al-Shaalan T M, Klie H, Dogru A H, et al. Studies of robust two stage preconditioners for the solution of fully implicit multiphase flow problems. Ppaer SPE 118722 presented at the SPE Reservoir Simulation Symposium, Woodlands, TX, USA, 2009

Appleyard J R, Cheshire I M and Pollard R K. Special techniques for fully implicit simulators. Proc. European Symposium on enhanced oil recovery, Bournemouth, England, 395-408, 1981

Appleyard J R and Cheshire I M. Nested factorization. Paper SPE 12264 presented at Proc. 7th SPE Symposium on Reservoir Simulation, 1983

Bank R E, Chan T F, Coughran J W M, et al. The alternate-block-factorization procedure for systems of partial differential equations. BIT. 1989. 29(4): 938-954

Behie A and Forsyth P A Jr. Incomplete factorization methods for fully implicit simulation of enhanced oil recovery. SIAM J. Sci. Stat. Comp. 1984. 5: 543-561

Behie G and Vinsome P. Block iterative methods for fully implicit reservoir simulation. Soc. Pet. Eng. J. 1982. 22(5): 658-668

Bjørstad P E, Manne F, Sørevik T, et al. Efficient matrix multiplication on SIMD computers. SIAM J. Matrix Anal. Appl. 1992. 13(1): 386-401

Brandt A, McCormick S and Ruge J. Algebraic Multigrid (AMG) for Sparse Matrix Equations, Sparsity and Its Applications. Cambridge Univ. Press, Cambridge. 1985. 257-284

Chen Z, Huan G and Ma Y. Computational methods for multiphase flows in porous media. Society for Industrial Mathematics, 2006

Christie M A and Blunt M J. Tenth SPE comparative solution project: a comparison of upscaling techniques. SPE Reservoir Evaluation & Engineering. 2001. 4: 308-317 (paper SPE 72469)

Concus P, Golub G H and Meurant G. Block preconditioning for the conjugate gradient method. SIAM J. Sci. Stat. Comput. 1985. 6: 220-252

Dogru A, Fung L, Middya U, et al. A next-generation parallel reservoir simulator for giant reservoirs. Paper SPE 119272 presented at SPE Reservoir Simulation Symposium, 2009

Douglas J Jr, Peaceman D W and Rachford H H Jr. A method of calculating multi-dimensional immiscible displacement. SPE AIME. 1959. 216: 297-396

Dupont T, Kendall R P and Rachford H H Jr. An approximate factorization procedure for solving self-adjoint elliptic difference equations. SIAM J. Numer. Anal. 1968. 5: 559-573

Falgout R. An introduction to algebraic multigrid. Computing in Science and Engineering. 2006. 8: 24-33

Feng C, Shu S and Yue X. An improvement for the OpenMP version BoomerAMG. Proceedings of CCF HPC China 2012, Zhangjiajie, China. 2012. 321-328

Feng C, Shu S, Xu J, et al. A multi-stage preconditioner for the black oil model and its OpenMP implementation. 21st International Conference on Domain Decomposition Methods (2012, INRIA Rennes-Bretagne-Atlantique), in LNCSE, Springer Berlin Heidelberg, 2014. 127-138

Feng C. Multilevel Iterative Methods and Solvers for Reservoir Simulation on CPU-GPU Heterogenous Computers. Ph.D. Thesis, Xiangtan University, Hunan, China, 2014

Han D K. The achievements and challenges of EOR technology for onshore oil fields in China. Proceedings of the 15th World Petroleum Congress, 363-372, 1998

Han D K, Yang C Z, Zhang Z Q, et al. Recent development of enhanced oil recovery in China. Journal of Petroleum Science and Engineering. 1999. 22: 181-188

Hayder M E and Baddourah M. Challenges in high performance computing for reservoir simulation. Ppaer SPE 152414 presented at the EAGE Annual Conference & Exhibition incorporating SPE Europec, Copenhagen, Denmark, 4-7, June 2012

Hu X, Liu W, Qin G, et al. Development of a fast auxiliary subspace preconditioner for numerical reservoir simulators. Paper SPE 148388 presented at SPE Reservoir Characterization and Simulation Conference, 2011

Hu X, Wu S H, Wu X H, et al. Combined preconditioning with applications in reservoir simulation. SIAM Multiscale Modeling and Simulation. 2013a. 11: 507-521

Hu X, Xu J and Zhang C S. Application of auxiliary space preconditioning in field-scale reservoir simulation. Science China Mathematics. 2013b. 56: 2737-2751

Hypre: A scalable linear solver library. URL: http://acts.nersc.gov/hypre/

Lacroix S, Vassilevski Y and Wheeler M. Decoupling preconditioners in the implicit parallel accurate reservoir simulator (IPARS). Numer. Linear Algebra with Applications. 2001. 8: 537-549

Lacroix S, Vassilevski Y, Wheeler J, et al. Iterative solution methods for modeling multiphase flow in porous media fully implicitly. SIAM J. Sci. Comput. 2003. 25: 905-926

Lam M D, Rothberg E E and Wolf M E. The cache performance and optimizations of blocked algorithms. Proceedings of the Fourth International Conference on Architectural Support for Programming Languages and Operating Systems (Asplos Iv), 1991. 63-74

Li Q Y, Wu S H, Wang B H, et al. A new generation reservoir simulator and its application in a matrure water-flooding oilfield. Paper SPE 166667 prensented at the Asia Pacific Oil & Gas Conference and Exhibition, Jakarta, Indonesia, 2013a

Li X B, Wu S H, Li Q Y, et al. An improved approach to simulate low-permeability fractured reservoir with dynamic hybrid dual-porosity model. Paper SPE166665 prensented at the Asia Pacific Oil & Gas Conference and Exhibition, Jakarta, Indonesia, 2013b

Meijerink J A and van der Vorst H A. An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix. Math. Comp. 1977. 31: 148-162

Meijerink J A. Iterative methods for the solution of linear equations based on the incomplete block factorization of the Matrix. Paper SPE12262 presented at the SPE Reservoir Simulation Symposium, Lubbock, TX. Nov. 14-15, 1983

Oliker L, Li X, Husbands P, et al. Effects of ordering strategies and programming paradigms on sparse matrix computations. SIAM Review. 2002. 44(3): 373-393

Pavlas E J Jr. Fine-scale simulation of complex water encroachment in a large carbonate reservoir in Saudi Arabia. SPE Reservoir Evaluation & Engineering. 2002. 5(5): 346-354 (paper SPE 79718)

Ruge J and Stüben K. Algebraic multigrid, in multigrid methods. In: Frontiers Appl. Math. Vol. 3, 73-130. SIAM, Philadelphia, PA, 1987

Saad Y. Iterative Methods for Sparse Linear Systems. Society for Industrial and Applied Mathematics, 2003

Stüben K. An introduction to algebraic multigrid. In: Trottenberg U, Oosterlee C and Schüller A. Multigrid. Academic Presss. 2001. 413-532

Stüben K, Clees T, Klie H, et al. Algebraic multigrid methods (AMG) for the efficient solution of fully implicit formulations in reservoir simulation. Paper SPE 105832 presented at the SPE Reservoir Simulation Symposium, Houston, TX, USA, 2007

Trangenstein J A and Bell J B. Mathematical structure of the black-oil model for petroleum reservoir simulation. SIAM Journal on Applied Mathematics. 1989. 49: 749-783

Vuduc R. Automatic Performance Tuning of Sparse Matrix Kernels. Ph.D. Thesis. University of California, Berkeley, CA, USA, 2003

Wallis J R. Incomplete Gaussian elimination as a preconditioning for generalized conjugate gradient acceleration. Paper SPE 12265 presented at the SPE Reservoir Simulation Symposium, San Francisco, California, November 15-18, 1983

Wallis J R, Kendall R P and Little T E. Constrained residual acceleration of conjugate residual methods. Paper SPE 13536 presented at the SPE Reservoir Simulation Symposium, Dallas, TX, February 10-13, 1985

Wang B H, Wu S H, Han D K, et al. Block compressed storage and computation in the large-scale reservoir simulation. Petroleum Exploration and Development. 2013a. 40: 495-500 (inChinese)

Wang B H, Wu S H, Li Q Y, et al. Applications of BILU0-GMRES in reservoir numerical simulation. ACTA Petrolei Sinica. 2013b. 34: 954-958 (inChinese)

Wang F and Xu J. A crosswind block iterative method for convection-dominated problems. SIAM Journal on Scientific Computing. 1999. 21: 620-645

Watts J W and Shaw J S. A new method for solving the implicit reservoir simulation matrix equation. Paper SPE 93068 presented at the SPE Reservoir Simulation Symposium, Texas, TX, USA, 2005

Wu S H, Xu J, Zhang C S, et al. Multilevel preconditioners for a new generation reservoir simulator. Paper SPE 166011 presented at SPE Reservoir Characterisation and Simulation Conference and Exhibition held in Abu Dhabi, UAE 2013a

Wu S H, Li X B, Li Q Y, et al. A dynamic hybrid model to simulation fractured reservoirs. Paper IPTC 16521 presented at the International Petroleum Technology Conference, Beijing, China, 2013b

(Edited by Sun Yanhua)