Journal of Computational and Applied Mathematics 328 (2018) 443–468

Contents lists available at ScienceDirect

Journal of Computational and Applied Mathematics

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

Efficient CPR-type preconditioner and its adaptive strategies for large-scale parallel reservoir simulations

Kun Wang *, Hui Liu, Jia Luo, Zhangxin Chen

Department of Chemical and Petroleum Engineering, University of Calgary, Canada

ARTICLE INFO

Article history:
Received 15 March 2016
Received in revised form 31 March 2017

ABSTRACT

With the rapid development of high performance parallel computers and the increasing demands of large-scale reservoir simulation, the development of effective and fast solvers and preconditioners for parallel reservoir simulation has become increasingly important. The incomplete LU factorization (ILU) preconditioners are the most commonly used preconditioners in reservoir simulations due to their low computational cost. They have been implemented in commercial simulators as the default preconditioners. However, for reservoirs with highly heterogeneous permeability, the CPR (constrained pressure residual)-type preconditioners are more effective than the ILU preconditioners, especially when parallel computing is employed. The CPR-type preconditioners consist of multi-stages of preconditioning processes, which lead to huge computational costs. Employing the algebraic multigrid (AMG) method to solve a pressure system and the ILU method to solve the whole system has two indispensable stages. For parallel computation, due to the serial nature of the ILU method, the ILU method is usually combined with the restricted additive Schwarz (RAS) method and used as a subdomain solver. Different settings of the AMG and RAS methods, such as a grid coarsening strategy, interpolation and smoothing in the AMG method, and a subdomain solver and an overlap level in the RAS method, as well as different CPR-type preconditioners, yield significantly different performance. This paper first gives an overview of the CPR-type preconditioners and the AMG and RAS methods. Then a detailed comparison between different CPR-type preconditioners with different settings of the AMG and RAS methods is presented. Based on the comparison, the efficient settings of the AMG and RAS methods and the most efficient CPR-type preconditioner are given. Furthermore, an adaptive preconditioning strategy is designed to save the computational time. The adaptive preconditioning strategy introduces an indicator to measure the difficulty level of solving a linear system. According to difficulties, a RAS–ILU preconditioner or a CPR-type preconditioner is automatically and dynamically selected to achieve both efficiency and effectiveness.

1. Introduction

Due to the requirement of a high-resolution description of complex geology, petroleum reservoir simulators often contain millions of grid cells or even more, and it may take days or even weeks for a simulator to complete one simulation run. According to [1], for reservoir simulation with a number of grid cells of order 100,000, about 80%–90% of the total simulation time is spent on the solution of linear systems resulting from discretizing the nonlinear equations of the black oil model,

* Corresponding author.
E-mail addresses: wangkun8576@gmail.com, wang30@ucalgary.ca (K. Wang).

http://dx.doi.org/10.1016/j.cam.2017.07.022
0377-0427/

because these systems are highly nonsymmetric and ill-conditioned. The Krylov subspace iterative methods, such as GMRES (Generalized Minimal Residual) and BiCGstab (Bi-conjugate Gradient Stabilized), are usually employed as linear solvers for the solution of these linear systems. However, these solvers do not yield a good convergence rate without a proper preconditioner. On the other hand, for parallel computing, direct parallelization of a classical serial preconditioner may reduce its efficiency, which makes the convergence of a linear solver even worse. The number of linear iterations can increase as more MPI processors are involved in computations. As a consequence, computing time cannot be reduced even if enough computational resources are available.

The development of multilevel (multigrid) methods is important in effective and fast solution of linear systems arising from the discretization of partial differential equations, particularly elliptic and parabolic equations. The algebraic multigrid method (AMG) [2,3] is one of the multilevel methods and a very efficient and scalable solver for solving positive-definite linear systems. For a linear system from an elliptic or parabolic equation discretized by the finite element (volume or difference) method, it can be solved by AMG to a given accuracy with computational work proportional to the system size. Two main processes of AMG are smoothing and coarse-grid correction. Normally, an iterative method, such as the Jacobi and Gauss–Seidel methods, is used in the smoothing process. For the coarse-grid correction, the residual of a linear system on a fine grid is restricted to a coarse grid. Then the linear system on the coarse grid is solved and its solution is interpolated back to the fine grid to correct an error on this grid. Many components of the AMG method can be parallelized in a straightforward way but some are highly sequential, such as its coarsening strategy and smoothers. Therefore, parallelization for these highly sequential components must be carefully implemented, since the quality of their parallelization affects the whole performance of the AMG method. Various parallel coarsening strategies [4–8], interpolations [9,10] and smoothers [11,12] have been developed. The performance of these methods for 2D and 3D elliptic problems has been tested in [6].

The domain decomposition methods [13–17] are another type of popular solvers or preconditioners in parallel computations. The additive Schwarz (AS) method is one of the domain decomposition methods. It was first introduced as a preconditioner for solving symmetric elliptic problems and then extended to the solution of nonsymmetric and non-elliptic problems. The restricted additive Schwarz (RAS) method [17] is a modification of the additive Schwarz method. Compared with AS as a preconditioner, the RAS preconditioner saves communication and leads to better convergence. In addition to elliptic problems, the RAS method has been successfully applied to a wide range of other problems, including convection–diffusion, Helmholtz's and Euler's problems.

For reservoir simulation, point-wise and block-wise incomplete factorizations [18,19] are often employed as the default preconditioners in commercial reservoir simulators. However, these preconditioners are not efficient enough for large-scale problems, especially for reservoir problems with highly heterogeneous permeability. The partial differential equations of the black oil model are parabolic with respect to pressure and hyperbolic with respect to saturation. Hence, from the physical and mathematical points of view, some more physical multi-stage preconditioners, such as constrained pressure residual (CPR) [20–25] and fast auxiliary space preconditioners (FASP) [26], have been developed. The main idea of these preconditioners is using different preconditioners to deal with the variables with different properties. In [20,21], the pressure system is solved by the AMG method as the first stage of the preconditioner, and an ILU process to the whole system is used in the second stage. In the FASP preconditioner [26], stages to deal with the well unknowns and the saturation unknowns are added. In [24], a family of CPR-type preconditioners are developed to improve efficiency by reordering the existing stages of the former preconditioners and adding new stages to them.

In this paper, we first apply the CPR-type preconditioners in large scale parallel reservoir simulations. Different parallel coarsening strategies, interpolations and smoothers in the AMG method, and different subdomain solvers and levels of overlaps in the RAS methods are compared in detail. According to the results, the efficient settings are chosen. With the efficient settings, different CPR-type preconditioners are compared, and the strategies of choosing the most efficient one according to a specific parallel reservoir simulation problem are given. In order to further improve the computational efficiency and save the computational time, an adaptive preconditioning strategy is designed. Depending on the difficulty of solving linear systems, the adaptive preconditioning strategy may choose and switch a preconditioner between the RAS–ILU preconditioner and the CPR-type preconditioner. The black oil simulator used in this paper is based on an in-house parallel platform we are developing, which is designed for general purpose simulators [25,27]. This platform is written in C and MPI (Message Passing Interface), and it provides grids, data, linear solvers, preconditioners, distributed matrices and vectors, visualization, key words parsing and well modelling modules. The RAS solver used in this paper is implemented in the platform. We use BoomerAMG from HYPRE [28], which is a library for solving large sparse linear systems of equations on massively parallel computers.

The rest of the paper is organized as follows: In Section 2, the mathematical equations of the black oil model and the numerical methods used to discretize these equations are presented. In Section 3, we briefly introduce the CPR-type preconditioners, AMG methods, and RAS method. In Section 4, numerical results with different settings in AMG and RAS are given, and then different CPR-type preconditioners are compared in detail. According to the numerical results, we obtain the efficient settings of the AMG and RAS methods, as well as the most efficient CPR-type preconditioner for a particular problem. In Section 5, an adaptive preconditioning strategy is designed to improve the efficiency of the CPR-type preconditioners. Finally, we give conclusions in Section 6.

2. Black oil model and numerical discretization

The classical black oil model assumes that the flow in a reservoir has three phases and three components (oil, gas and water). Combining Darcy's law and mass conservation law, the black oil model is written as follows [1,29]:

$$\begin{cases} \dfrac{\partial}{\partial t}(\phi s_o \rho_o^o) = \nabla \cdot \left(\dfrac{KK_{ro}}{\mu_o} \rho_o^o \nabla \Phi_o \right) + q_o, \\ \dfrac{\partial}{\partial t}(\phi s_w \rho_w) = \nabla \cdot \left(\dfrac{KK_{rw}}{\mu_w} \rho_w \nabla \Phi_w \right) + q_w, \\ \dfrac{\partial(\phi \rho_o^g s_o + \phi \rho_g s_g)}{\partial t} = \nabla \cdot \left(\dfrac{KK_{ro}}{\mu_o} \rho_o^g \nabla \Phi_o \right) + \nabla \cdot \left(\dfrac{KK_{rg}}{\mu_g} \rho_g \nabla \Phi_g \right) + q_o^g + q_g, \\ \Phi_\alpha = p_\alpha + \rho_\alpha \wp z, \\ s_o + s_w + s_g = 1, \\ p_w = p_o - p_{cow}, \\ p_g = p_o + p_{cog}, \end{cases} \tag{1}$$

where ϕ and K are porosity and permeability, for phase α ($\alpha = o, w, g$), Φ_α is the phase potential, and s_α, μ_α, p_α, ρ_α, $K_{r\alpha}$ and q_α are the saturation, viscosity, pressure, density, relative permeability and production rate, respectively. p_{cow} and p_{cog} are the oil–water and oil–gas capillary pressures. ρ_o^g is the density of the solution gas, ρ_o^o is the density of the oil component, and $\rho_o = \rho_o^g + \rho_o^o$. $\wp$ is the gravitational constant and z is the reservoir depth. These variables have the following relations:

$$\begin{aligned} &\phi = \phi(p_o), \\ &K_{ro} = K_{ro}(s_w, s_g), \\ &K_{rw} = K_{rw}(s_w), \\ &K_{rg} = K_{rg}(s_g), \\ &\rho_o^o = \rho_o^o(p_o, p_b), \\ &\rho_o^g = \rho_o^g(p_o, p_b), \\ &\rho_w = \rho_w(p_w), \\ &\rho_g = \rho_g(p_g), \\ &\mu_o = \rho_o(p_o, p_b), \\ &\mu_w = \rho_w(p_w), \\ &\mu_g = \rho_g(p_g), \\ &p_{cow} = p_{cow}(s_w), \\ &p_{cog} = p_{cog}(s_g), \end{aligned}$$

where p_b is the bubble point pressure. With proper boundary and initial conditions, a close system is given.

In this paper, the fully implicit method (FIM) in time and a first-order upstream finite difference method in space are applied to discretize the black oil model, where pressure p_o, water saturation s_w and X (gas saturation s_g or bubble point pressure p_b) are chosen as unknowns. By introducing the transmissibility term

$$\begin{aligned} T_{o,d} &= \frac{KK_{ro}}{\mu_o} \rho_o^o \frac{A}{\Delta d} \\ T_{w,d} &= \frac{KK_{rw}}{\mu_w} \rho_w \frac{A}{\Delta d} \\ T_{g,d} &= \frac{KK_{rg}}{\mu_g} \rho_g \frac{A}{\Delta d} \\ T_{og,d} &= \frac{KK_{ro}}{\mu_o} \rho_o^g \frac{A}{\Delta d} \end{aligned}$$

and defining the term

$$\begin{aligned} \Delta T_\alpha \Delta \Phi_\alpha \equiv\ & (T_{\alpha,x})_{i+\frac{1}{2},j,k}(\Phi_{i+1,j,k} - \Phi_{i,j,k}) - (T_{\alpha,x})_{i-\frac{1}{2},j,k}(\Phi_{i-1,j,k} - \Phi_{i,j,k}) \\ & + (T_{\alpha,y})_{i,j+\frac{1}{2},k}(\Phi_{i,j+1,k} - \Phi_{i,j,k}) - (T_{\alpha,y})_{i,j-\frac{1}{2},k}(\Phi_{i,j-1,k} - \Phi_{i,j,k}) \\ & + (T_{\alpha,z})_{i,j,k+\frac{1}{2}}(\Phi_{i,j,k+1} - \Phi_{i,j,k}) - (T_{\alpha,z})_{i,j,k-\frac{1}{2}}(\Phi_{i,j,k-1} - \Phi_{i,j,k}), \end{aligned} \tag{2}$$

we can discretize and linearize the nonlinear system (1) as

$$\begin{cases} \frac{V}{\Delta t}[(\phi\rho_o^o s_o)^l - (\phi\rho_o^o s_o)^n + \delta(\phi\rho_o^o s_o)] = \Delta(T_o^l + \delta T_o)\Delta(\Phi_o^l + \delta\Phi_o) + q_o^l + \delta q_o \\ \frac{V}{\Delta t}[(\phi\rho_w s_w)^l - (\phi\rho_w s_w)^n + \delta(\phi\rho_w s_w)] = \Delta(T_w^l + \delta T_w)\Delta(\Phi_w^l + \delta\Phi_w) + q_w^l + \delta q_w \\ \frac{V}{\Delta t}[(\phi\rho_o^g s_o + \phi\rho_g s_g)^l - (\phi\rho_o^g s_o + \phi\rho_g s_g)^n + \delta(\phi\rho_o^g s_o + \phi\rho_g s_g)] \\ \quad = \Delta(T_{og}^l + \delta T_{og})\Delta(\Phi_o^l + \delta\Phi_o) + \Delta(T_g^l + \delta T_g)\Delta(\Phi_g^l + \delta\Phi_g) + q_{og}^l + q_g^l. \end{cases} \tag{3}$$

Then, a linear system

$$\begin{pmatrix} A_{ws_w} & A_{wX} & A_{wp} \\ A_{os_w} & A_{oX} & A_{op} \\ A_{gs_w} & A_{gX} & A_{gp} \end{pmatrix} \begin{pmatrix} s_w \\ X \\ p_o \end{pmatrix} = \begin{pmatrix} b_w \\ b_o \\ b_g \end{pmatrix} \tag{4}$$

is obtained from (3), where matrices $A_{\alpha s_w}$, $A_{\alpha X}$ and $A_{\alpha p}$ ($\alpha = o, w, g$) are the submatrices corresponding to water saturation s_w, gas saturation s_g (or bubble point pressure p_b) and pressure p_o, respectively. The linear system (4) can be denoted by

$$Ax = b. \tag{5}$$

3. Parallel preconditioners for reservoir simulations

3.1. Multi-stage preconditioners

In practice, an equivalent system, $M^{-1}Ax = M^{-1}b$, is solved, where a preconditioner M is used to accelerate convergence. From the view of mathematics, the pressure equation from the black oil model is elliptic, while the equations are hyperbolic corresponding to the saturation or bubble point pressure, if the capillary pressure is ignored. To deal with the submatrices from (4) with different properties separately, multi-stage preconditioners are developed, which include the constrained pressure residual (CPR) [20,21], fast auxiliary space (FASP) [26] and CPR-type preconditioners [24,25]. The inverse of M can be described by a general formula. Let r^0 be the initial residual:

$$r^0 = b - Ax^0, \tag{6}$$

where x^0 is an initial guess. The residual r^n at the end of the nth-stage preconditioning process can be represented as

$$r^n = (I - AM_n^{-1})r^{n-1}. \tag{7}$$

Then the general formula is

$$M^{-1} = M_n^{-1}(I - AM_{n-1}^{-1})\cdots(I - AM_1^{-1}) + \cdots + M_2^{-1}(I - AM_1^{-1}) + M_1^{-1}, \tag{8}$$

where M_i^{-1} is the ith stage preconditioner.

Before the preconditioning process, a decoupling operation, such as the Quasi-IMPES [30], Householder transformation [31], and alternative block factorization (ABF) methods [32], should be applied to the linear system, which can weaken the strong coupling between the pressure unknowns and saturation or bubble point pressure unknowns. We choose the ABF method in our computation due to its effectiveness and inexpensive computation, then the linear system (4) can be written as

$$\tilde{A}x = \tilde{b}, \tag{9}$$

where $\tilde{A} = D^{-1}A$, $\tilde{b} = D^{-1}b$, and

$$D = \begin{pmatrix} \mathbf{Diag}(A_{ws_w}) & \mathbf{Diag}(A_{wX}) & \mathbf{Diag}(A_{wp}) \\ \mathbf{Diag}(A_{os_w}) & \mathbf{Diag}(A_{oX}) & \mathbf{Diag}(A_{op}) \\ \mathbf{Diag}(A_{gs_w}) & \mathbf{Diag}(A_{gX}) & \mathbf{Diag}(A_{gp}) \end{pmatrix}.$$

Here $\mathbf{Diag}(\cdot)$ stands for a diagonal matrix. To make the notation simple, we will still use $Ax = b$ to represent the decoupled system $\tilde{A}x = \tilde{b}$.

The well-known CPR method is a two-stage preconditioner, which separates the pressure block from the full system and uses the AMG method to solve the pressure block at the first stage and then a global smoother (such as the ILU method) to the full system at the second stage. In order to describe the CPR method, we introduce a transfer operator Π_p in the following formula:

$$\Pi_p \begin{pmatrix} b_w \\ b_o \\ b_g \end{pmatrix} = b_g. \tag{10}$$

Then the CPR preconditioner can be presented in Algorithm 1.

Algorithm 1 CPR Preconditioner

1: Given an initial guess x^0, find
2: $x^1 = x^0 + \Pi_p M_p^{-1} \Pi_p^T (b - Ax^0)$,
3: $x^2 = x^1 + R^{-1}(b - Ax^1)$.

In the first stage, M_p^{-1} is designed to solve the pressure submatrix A_{gp} with the AMG method, which is efficient for an elliptic problem. Usually, one AMG V-cycle is enough for the preconditioning process. The preconditioner R in the second stage solves the full system. The LSOR [26,33], ILU (or block ILU) and block Gauss–Seidel methods have been successfully applied in serial computations. However, for parallel computations, these methods are normally not used directly, since their serial nature affects the parallel scalability. Hence they are usually combined with the domain decomposition methods, which means that they are only employed as a subdomain solver, not a global one. The restricted additive Schwarz (RAS) method [17] is one of the most widely used domain decomposition methods, which is easier to parallelize and has less computation time and fewer iterations than other domain decomposition methods. For simplicity, we use RAS–ILU to represent the RAS method with the ILU method as the subdomain solver.

Besides the two stages in the CPR preconditioner, the FASP preconditioner involves two extra stages, which are to deal with the well unknowns and the saturation unknowns. The subsystem corresponding to the well unknowns is solved by direct methods. However, for parallel reservoir simulations, the subsystem should be gathered into one MPI process, which significantly affects the parallel efficiency. Thus, this stage will not be considered in this paper. Another extra stage is to solve the subsystem corresponding to the saturation unknowns. Similar as the operator Π_p, a transfer operator Π_S is defined as:

$$\Pi_S \begin{pmatrix} b_w \\ b_o \\ b_g \end{pmatrix} = \begin{pmatrix} b_w \\ b_o \end{pmatrix}. \tag{11}$$

The FASP preconditioner can be represented in Algorithm 2. In this paper, the RAS–ILU method is employed at the first stage of the FASP preconditioner, which is represented as R_S^{-1}.

Algorithm 2 The FASP Preconditioner

1: Given an initial guess x^0, find
2: $x^1 = x^0 + \Pi_S R_S^{-1} \Pi_S^T (b - Ax^0)$,
3: $x^2 = x^1 + \Pi_p M_p^{-1} \Pi_p^T (b - Ax^1)$,
4: $x^3 = x^2 + R^{-1}(b - Ax^2)$.

In [23–25], the stage dealing with the saturation unknowns is replaced by a RAS–ILU process to the global system, which results in a new CPR-type preconditioner. We denote this preconditioner by CPR-FPF and represent it in Algorithm 3. Compared to the FASP preconditioner, the CPR-FPF preconditioner saves the setup time of the incomplete LU factorization for the saturation subsystem, but brings extra computation in the iterative process. However, the extra RAS–ILU stage makes the CPR-FPF preconditioner more effective than the CPR and FASP preconditioners. A four stage preconditioner is also presented in [23], which can be represented in Algorithm 4.

Algorithm 3 The CPR-FPF Preconditioner

1: Given an initial guess x^0, find
2: $x^1 = x^0 + R^{-1}(b - Ax^0)$,
3: $x^2 = x^1 + \Pi_p M_p^{-1} \Pi_p^T (b - Ax^1)$,
4: $x^3 = x^2 + R^{-1}(b - Ax^2)$.

Algorithm 4 The CPR-FFPF Preconditioner

1: Given an initial guess x^0, find
2: $x^1 = x^0 + R^{-1}(b - Ax^0)$,
3: $x^2 = x^1 + R^{-1}(b - Ax^1)$,
4: $x^3 = x^2 + \Pi_p M_p^{-1} \Pi_p^T (b - Ax^2)$,
5: $x^4 = x^3 + R^{-1}(b - Ax^3)$.

3.2. Algebraic multigrid method

The multigrid method is one of multilevel methods, which can solve linear systems from the discretization of elliptic and parabolic partial differential equations efficiently and scalability. Compared with the geometric multigrid method, which uses a geometry to define a series of subgrids, the algebraic multigrid (AMG) only uses the information in linear systems, which makes it more convenient for the user and more suitable for a problem with a very complicated domain. Many different kinds of coarsening strategies, interpolation procedures and smoothing methods have been developed to achieve the convergence. Different methods lead to different operation complexity, which affects the setup time, cycle time and convergence rate. Less complexity, which can reduce the setup and cycle times, may lead to degradation in convergence, and increasing complexity may lead to better convergence but require more time in the setup and solution phases. For AMG we must decide its priorities to achieve both minimal computational time and parallel scalability. For the coarsening strategies, we will consider the RS3 [6], CLJP (Cleary–Luby–Jones–Plassman) [7], FALGOUT [6], PMIS and HMIS [8] strategies. According to different coarsening strategies, many interpolation operators are developed. A modified classical interpolation [4], the standard interpolation, a multipass interpolation and a modified extended interpolation [9] are discussed in this paper. A smoother also plays an important role in the AMG method, and a good one can reduce an oscillatory error. The C–F Jacobi [6] smoother, hybrid Gauss–Seidel smoother [11], and ℓ_1-scaled smoothers [12] are studied.

3.3. Restricted additive Schwarz method

The domain decomposition method has been widely used as preconditioner in parallel computing due to its parallel nature. The additive Schwarz method (AS) was first introduced for solving symmetric positive-define linear systems, and then extended to many other nonsymmetric systems [13–16]. Cai and Sarkis modified the AS preconditioner and developed the restricted additive Schwarz (RAS) preconditioner [17]. The RAS preconditioner can save half of the communication cost, since no data exchange with the neighbouring processors is involved after the subdomain problems are solved. Moreover, the RAS preconditioner has better performance with fewer iterations and less computation time. As a preconditioner, the subdomain problems are usually not solved accurately. Inexact solvers are employed to save computational costs, as well as to achieve a reasonable convergence rate. ILU (Incomplete Factorization)-like methods, such as the ILU(k) and ILUT methods, are popular subdomain solvers. Normally, the more exactly the subdomain problems are solved, the fewer iterations are needed to achieve the desired accuracy, but the more computational time may be resulted in. The level δ of overlap is another important factor for the performance of the RAS preconditioner. With the same situation as in the choice of subdomain solvers, a large number of overlap levels improves convergence but leads to more computational costs, especially when the number of processors is large. A small number of overlap levels saves computational costs but may lead to worse convergence. The user must decide his/her priorities.

4. Efficiency tests of the CPR-type preconditioners

The black oil SPE10 case and the refined black oil SPE10 case, which will be introduced in Sections 4.1 and 4.2, are tested on the cluster Parallel from Westgrid and GPC (General Purpose Cluster) from Canada's largest supercomputer centre SciNet, respectively. Parallel has 7,056 CPU cores, 528 12-core standard nodes and 60 special 12-core nodes that have three general-purpose GPUs each. The 12 cores associated with one compute node share 24 GB of RAM. Parallel uses an InfiniBand 4X QDR (Quad Data Rate) 40 Gbit/s switched fabric, with a two to one blocking factor. GPC consists of 3780 nodes (IBM iDataPlex DX360M2) with a total of 30,240 cores (Intel Xeon E5540) at 2.53 GHz, with 16 GB RAM per node (some larger-memory nodes up to 32 GB). Our jobs were run on the nodes with 16 GB memory connected with non-blocking DDR InfiniBand. The memory and cores of each employed node in Parallel and GPC are fully used in our computation, and each MPI process runs on a core.

In the numerical experiments, the time step is self-adaptive and controlled by the following formula

$$t^n = \sigma^{n-1} t^{n-1}, \tag{12}$$

where t^n and t^{n-1} are the nth and $(n-1)$-th time step, and

$$\sigma^{n-1} = \min\left\{ \frac{\delta p^{n-1}}{\delta p_{\max}}, \frac{\delta s_w^{n-1}}{\delta s_{w,\max}}, \frac{\delta s_g^{n-1}}{\delta s_{g,\max}}, \sigma_{\max} \right\}, \tag{13}$$

p^{n-1}, s_w^{n-1}, s_g^{n-1} are the maximal change of pressure, water saturation and gas saturation among all the grid blocks at the $(n-1)$-th time step, respectively; $p_{\max}, s_{w,\max}, s_{g,\max}$ are the expected change of pressure, water saturation and gas saturation at each time step, respectively; and $\sigma_{\max}$ is the maximal increasing rate. In this paper, we set $p_{\max} = 1000$psi, $s_{w,\max} = 0.05$, $s_{g,\max} = 0.05$, and $\sigma_{\max} = 3.0$. More details about this adaptive technique can be found in [1].

The inexact Newton method is employed as the nonlinear method to avoid oversolving the linear systems and save computational time. The required relative residual θ_l for the linear system resulted from the lth Newton iteration is calculated

as

$$\theta_l = \gamma \left(\frac{b^l}{b^{l-1}} \right)^{\beta}, \tag{14}$$

where $\gamma = 0.5$ and $\beta = \frac{\sqrt{5}+1}{2}$. A safeguard for θ_l is also used as follows,

$$\theta_l = \begin{cases} 0.05, & \text{if} \quad \theta_l > 0.05 \\ \theta_l, & \text{if} \quad 0.05 < \theta_l < 0.005 \\ 0.005, & \text{if} \quad \theta_l < 0.005. \end{cases} \tag{15}$$

More details about the inexact Newton method can be found in [25,34]. The stopping criteria of Newton iterations used in the numerical experiments can be represented as follows:

- The scaled mass balance errors e_α on each grid block for the phase α ($\alpha = o, w, g$), which are defined as

$$e_o = \frac{\frac{V}{\Delta t}[(\phi \rho_o^o s_o)^{n+1} - (\phi \rho_o^o s_o)^n] - \Delta T_{o,o}^{n+1} \Delta \Phi_o^{n+1} - q_{o,o}^{n+1}}{\frac{V}{\Delta t}[(\phi \rho_o^o s_o)^{n+1} - (\phi \rho_o^o s_o)^n]}$$

$$e_w = \frac{\frac{V}{\Delta t}[(\phi \rho_w s_w)^{n+1} - (\phi \rho_w s_w)^n] - \Delta T_w^{n+1} \Delta \Phi_w^{n+1} - q_w^{n+1}}{\frac{V}{\Delta t}[(\phi \rho_w s_w)^{n+1} - (\phi \rho_w s_w)^n]}$$

$$e_g = \frac{\frac{V}{\Delta t}[(\phi \rho_o^g s_o)^{n+1} - (\phi \rho_o^g s_o)^n] - \Delta T_{o,g}^{n+1} \Delta \Phi_o^{n+1} - q_{o,g}^{n+1}}{\frac{V}{\Delta t}[(\phi \rho_o^g s_o + \phi \rho_g s_g)^{n+1} - (\phi \rho_o^g s_o + \phi \rho_g s_g)^n]} + \frac{\frac{V}{\Delta t}[(\phi \rho_g s_g)^{n+1} - (\phi \rho_g s_g)^n] - \Delta T_g^{n+1} \Delta \Phi_g^{n+1} - q_g^{n+1}}{\frac{V}{\Delta t}[(\phi \rho_o^g s_o + \phi \rho_g s_g)^{n+1} - (\phi \rho_o^g s_o + \phi \rho_g s_g)^n]},$$

  should satisfy

$$e_\alpha < \epsilon \quad \alpha = o, w, g \tag{16}$$

- The maximum saturation change e_s should satisfy

$$e_s < \max(0.1 \times \epsilon, 0.01). \tag{17}$$

- The scaled pressure increment of Newton iteration e_p on each grid block, which is defined as

$$e_p = \frac{\delta p}{p},$$

  should satisfy

$$e_p \leq \max(0.1 \times \epsilon, 0.01). \tag{18}$$

4.1. AMG and RAS tests

In this subsection, we will randomly select a time step and compare the performance of coarsening strategies, interpolations, smoothers and levels of coarse grids in the AMG method and the performance of the overlap levels and subdomain solvers in the RAS method. Since reservoir simulation is a complicated nonlinear time-dependent problem, the linear systems that resulted from Newton iterations vary significantly depending on many factors, such as the size of time step and well constraints. Therefore, we will also present the overall performance of the considered methods during the entire simulation. The CPR preconditioner is used as a representative in this test.

The Tenth SPE Comparative Project is a challenging test case for linear solvers due to its highly heterogeneous permeability and porosity. Its permeability varies between 6.65×10^{-11} and 20 Darcy, and its porosity varies between 0 and 0.5. The dimensions are 1200 ft $\times$ 2200 ft $\times$ 170ft. The reservoir has been divided into 1.122 million ($60 \times 220 \times 85$) of grid cells, and there are 3.366 million of unknowns in a linear system. The top 35 layers (70 ft) represent the Tarber formation and the bottom 50 layers (100 ft) represent the Upper Ness formation. Four producers are placed at four corners of the reservoir, and one injector in the centre. The total simulation time is 2000 days. More details about this case can be found in [35].

Fluid properties are adopted for a black oil problem; see Table 1. B_o is the oil formation volume factor, R_s is the solution gas–oil ratio for saturated oil and $C_o = 1.14 \times 10^{-6}$ psia^{-1} is the compressibility of undersaturated oil. The water formation volume factor is calculated by

$$B_w = \frac{B_w^0}{1 + C_w(p - p^0)}, \tag{19}$$

Table 1
Oil and Gas PVT Table.

Pressure psia	R_s MSCF/STB	B_o RB/STB	μ_o cp	B_g SCF/STB	μ_g cp
400	0.0165	1.012	3.5057	1.96	0.0140
4 000	0.1130	1.01278	2.9972	0.84	0.0160
8 000	0.1583	1.10759	2.7775	0.59	0.0175
10 000	0.1810	1.155	2.6675	0.42	0.0195

where the reference formation volume factor $B_w^0 = 1.01$, water compressibility $C_w = 3 \times 10^{-6}$ psia^{-1} and the reference pressure $p^0 = 6000$ psia. The water viscosity $\mu_w = 0.3$ cp is constant.

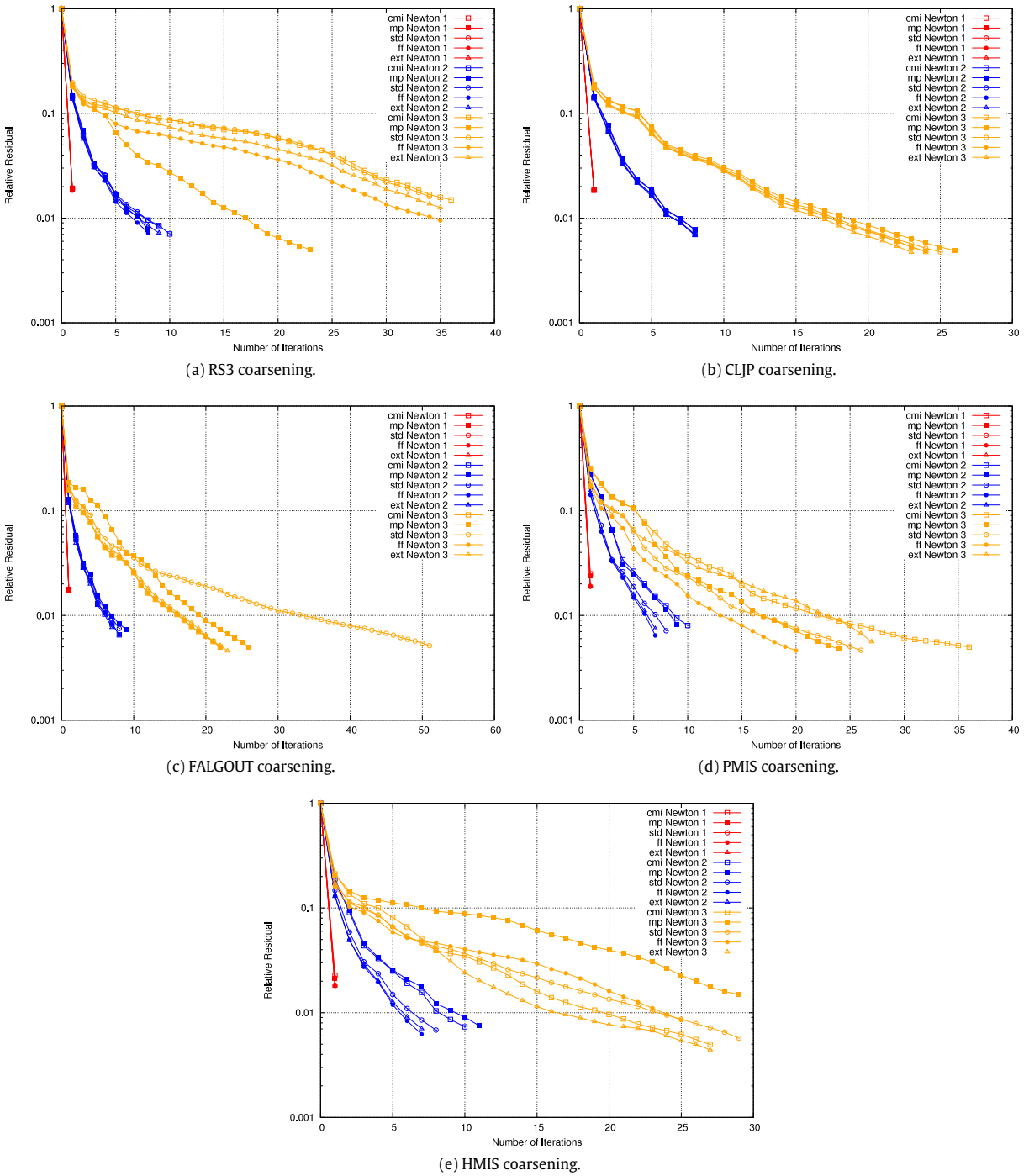
In this subsection, the stopping criterion of the nonlinear iteration ϵ in Eqs. (16)–(18) is set to be 10^{-2}. GMRES is employed as the linear solver and the CPR method is employed as a preconditioner. In Tables 2–8, "T.S". is the total time steps, "# N" is the total number of Newton iterations, "# L" is the total number of linear iterations, "A.L.N" is the average number of linear iterations per Newton iteration, "AMG" is the AMG setup time (in seconds), "L. Time" is the total linear solver time (including setup and solution time, in seconds) and "T. Time" is the total simulation time (in seconds). "C.S"., "Interp" and "Smoother" are the coarsening strategy, interpolation, and smoother in the AMG method, respectively. "cmi", "mp", "std", "ext" and "ff" are the modified classical interpolation, multipass interpolation, standard interpolation, extended interpolation and FF interpolation, respectively. "GS-F", "GS-B" and "GS-S" are the forward, backward and symmetric hybrid Gauss–Seidel smoothers, respectively. "Sub Solver" and "Overlap" are the subdomain solver and the level of overlap in the RAS method, respectively. Since this simulation case contains 1.1 million of grid cells, we use up to 144 MPI processes in order to use a reasonable number of grid cells in each processor.

First, we study the effect of the coarsening strategies and interpolation on parallel reservoir simulation. In this test, 144 MPI processes are employed, the number of coarse grid levels is set to six, the forward hybrid Gauss–Seidel is used as the smoother, the overlap of the RAS method is 0, and ILU(0) is used as the subdomain solver. The performance of coarsening strategies and interpolations in a time step with three Newton iterations is shown in Fig. 1(a)–(e). The number of non-zero elements in the linear systems is around 48,363,000 and slightly varies at each Newton iteration. We can see that the number of linear iterations is close when different types of interpolations are used with a specific coarsening strategy, except the combinations of RS3 coarsening strategy and multipass interpolation, FALGOUT coarsening strategy and standard interpolation, as well as PMIS coarsening strategy and classical interpolation. It can be explained that an interpolation method is only one part of the AMG method and the AMG process is only employed as one stage of the CPR preconditioner, then the quality of interpolation does not significantly affect the overall convergence of the CPR preconditioned linear solvers. In this situation, the time cost is an important factor we need to consider. From Fig. 2(a)–(c), we can see that the multipass interpolation has much less AMG setup time and the solving time of all the combinations is similar, therefore the total time of linear solver that employs the multipass interpolation is the least. Then the overall performance is summarized in Table 2, from which we could draw the similar conclusion. In general, the multipass interpolation that is based on the direct interpolation is not as powerful as others and the linear solver may use more iterations. However, only one or two more linear iterations per Newton iteration are employed. The advantage of the multipass interpolation is its fairly cheap computation cost. Benefiting from this advantage, the least AMG setup time and the least total linear solver time are used.

Once the interpolation method is chosen, all the tested coarsening strategies give similar performance. We re-test the performance of coarsening strategies with a larger problem using more MPI processes. In this larger problem, each grid block of the black oil SPE10 case is refined into 27 grid cells, which results in $180 \times 660 \times 255 = 30,294,000$ grid cells. The linear systems have $30,294,000 \times 3 = 90,882,000$ unknowns. Considering the size of this case, from 256 to 1024 MPI processes are used. Because of the huge computational resource required and the limitation of the simulation time, 300 time steps are run.

Again, for a selected time step, the convergence history of the linear solvers is plotted in Fig. 3(a), and the time cost is summarized in Fig. 3(b)–(d). The linear systems contain around 1.3 billion of none-zero elements. The PMIS and HMIS methods lead to fewer linear iterations, and they need less setup time and solving time. The overall performance is shown in Table 3. We can also see that the aggressive coarsening strategies, HMIS and PMIS methods, have better performance of both linear iteration times and computational time. Therefore, it can be concluded that the PMIS and HMIS coarsening strategies are more robust and efficient than the others.

Because the HMIS and PMIS coarsening strategies with the multipass interpolation have better performance than other combinations, we use them to test the effect of different smoothers. The convergence history of the linear solvers is shown in Fig. 4(a)–(b). When the PMIS coarsening strategy is used, the three hybrid Gauss–Seidel methods lead to fewer iterations than the C–F Jacobi and ℓ_1-Jacobi methods. The hybrid Gauss–Seidel methods also present robust performance when the HMIS coarsening strategy is employed. From the view of computational cost, see Fig. 5(a)–(b), the ℓ_1-Jacobi method is the worst choice while it is difficult to tell which one is better among the other four methods. We re-used the refined SPE10 case to test the smoothers; see Fig. 6(a)–(d). We can see that the C–F Jacobi leads to poor convergence rate and more computational time while the other four methods give similar performance. In Table 4, the overall performance of different smoothers is shown, from which we can see that the hybrid Gauss–Seidel methods used less computational time. From the above numerical



(a) RS3 coarsening.

(b) CLJP coarsening.

(c) FALGOUT coarsening.

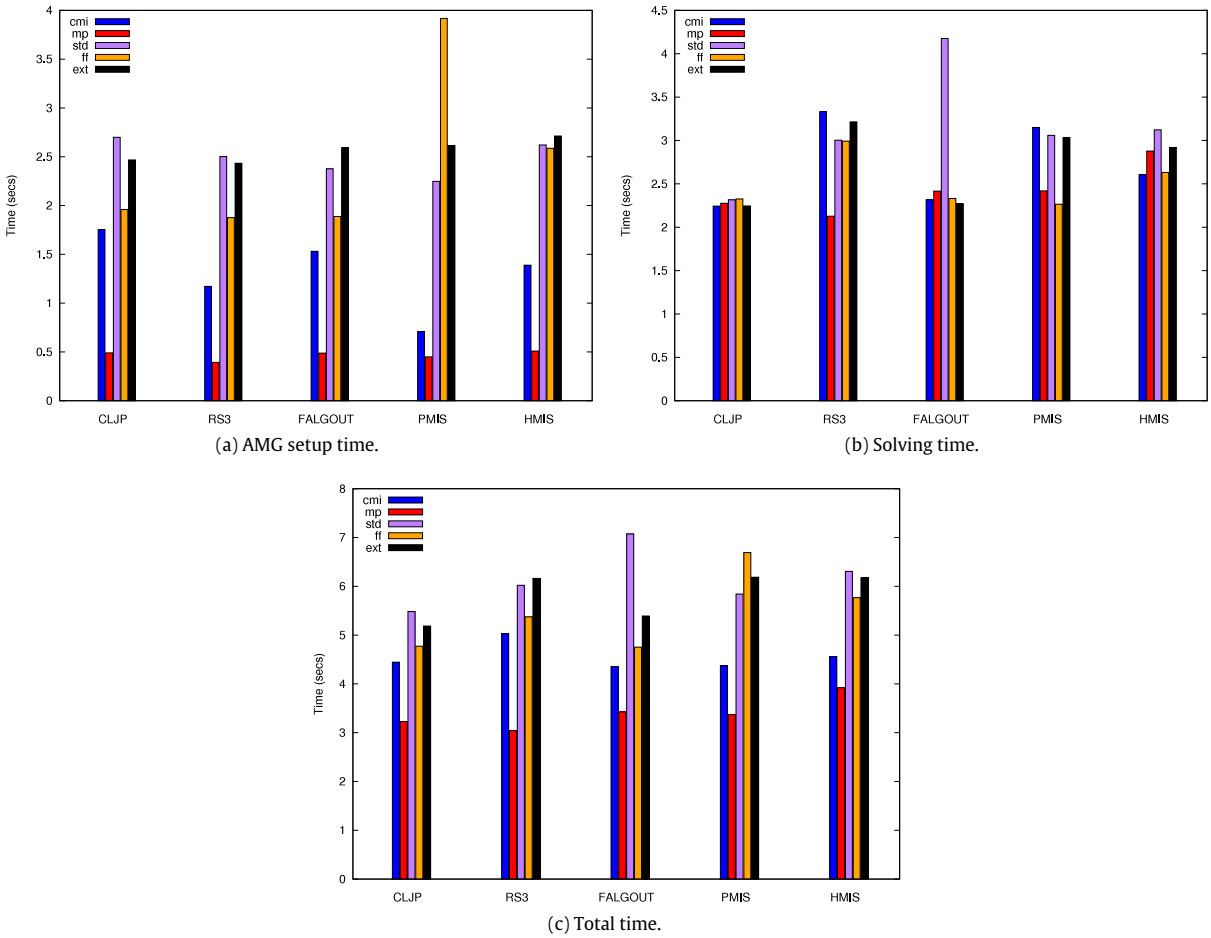
(d) PMIS coarsening.

(e) HMIS coarsening.

Fig. 1. Comparison of interpolations.

experiments, we can conclude that the hybrid Gauss–Seidel methods are more robust and have better performance than others.

The number of coarse grid levels is an another important factor affecting the performance of a preconditioner and parallel scalability. Learning from the above two test results, we use the HMIS and PMIS coarsening strategies, the multipass interpolations, and the forward hybrid Gauss–Seidel smoother in this test. We tested 3, 6 and 9 levels of coarse grids. It is clear that the number of linear iterations decreases when the number of coarse grid levels increases; see Fig. 7(a)–(b).



(a) AMG setup time.

(b) Solving time.

(c) Total time.

Fig. 2. Time cost of different interpolations.

However, the computational cost of extra coarse grid levels, including the AMG setup time and AMG V-cycle time, will increase dramatically; see Fig. 7(c)–(d). When 3 levels of coarse grids are used, the linear solver is not robust and the number of linear iterations is much larger. Although it has the least computational cost at each linear iteration, too many linear iterations make it not an efficient choice. For 9 levels of coarse grids, since the number of linear iterations does not substantially decrease as the increasing of coarse grids, the huge computational cost of each linear iteration makes it not an efficient choice either. According to the numerical experiments, 6 levels of coarse grids are a better choice, which performs good convergence rate and uses less expensive computational cost. In Table 5, we summarized the overall performance. We can see some similar phenomena (1) a larger number of coarse grid levels brings fewer linear iterations; (2) the extra computational cost required by the large number of coarse grids makes the nine levels of coarse grids the worst choice among all the tests; (3) the least computational time is achieved when six levels of coarse grids are used.

To test different subdomain solvers in the RAS method, we use the PMIS and HMIS coarsening strategies, the multipass interpolation, forward hybrid Gauss–Seidel smoother and 6 levels of coarse grids in the following tests. The subdomain solvers, ILU(0), ILU(1), ILU(2) and ILUT(p, τ), are tested. In the ILUT(p, τ) method, we set the threshold parameters $\tau = 0.001$ and

$$p = \frac{(N_{\text{row}} - 1 + N_{\text{nonzero}}) \times 1.5}{N_{\text{row}}}, \tag{20}$$

where N_{row} is the number of rows in a matrix and N_{nonzero} is the number of nonzero elements in the matrix. Compared with the ILU(k) methods, the ILUT method has robust performance and always leads to fewer linear iterations; see Fig. 8a and b. Besides that, the ILUT method also uses the least setup time and leads to the least solving time; see Fig. 8c and d. The ILU(0) method also has fewer setup time, but sometimes it leads to more linear iterations and more computational time. The extra levels of fill in the factorization of ILU(1) and ILU(2) do not significantly improve the efficiency of the linear solver, but increase the computational complexity. The overall performance is shown in Table 6. "ASS" is the total time of assembling

Table 2
SPE10 case: comparison of different coarsening strategies and interpolation, 144 MPI processes are employed, the number of coarse grid levels is 6, hybrid forward Gauss–Seidel is used as smoother, the overlap of RAS method is 0, and ILU(0) is used as the subdomain solver.

C.S.	Interp	T.S.	# N	# L	A.L.N	AMG	L. Time
	cmi	407	1378	52,825	38.3	634	3619
	mp	**407**	**1373**	**59,031**	**42.9**	**180**	**3306**
RS3	std	407	1354	50,557	37.3	1069	4025
	ext	407	1355	51,065	37.6	1063	4368
	ff	407	1350	50,288	37.2	876	3906
	cmi	407	1363	53,762	39.4	747	3790
	mp	**407**	**1369**	**53,176**	**38.8**	**212**	**3431**
CLJP	std	407	1361	51,772	38.0	1036	4005
	ext	407	1346	50,843	37.7	1021	4289
	ff	407	1363	53,762	39.4	867	4356
	cmi	407	1372	51,909	37.8	803	3751
	mp	**407**	**1371**	**59,625**	**43.5**	**226**	**3432**
FALGOUT	std	407	1374	50,990	37.1	1098	4067
	ext	407	1362	50,828	37.3	1091	4042
	ff	407	1372	51,909	37.8	950	3882
	cmi	407	1397	62,589	44.8	726	4379
	mp	**407**	**1417**	**55,080**	**38.8**	**185**	**3400**
PMIS	std	407	1424	44,725	31.4	1207	4433
	ext	407	1442	45,835	31.7	1243	5519
	ff	407	1441	45,957	31.8	1150	4353
	cmi	407	1394	59,699	42.8	730	4212
	mp	**407**	**1415**	**51,474**	**36.3**	**212**	**3168**
HMIS	std	407	1433	45,980	32.0	1264	4646
	ext	407	1433	44,063	30.7	1259	4508
	ff	407	1421	44,855	31.5	1162	4294

Table 3
Refined SPE10 case: comparison of different coarsening strategies, 1024 MPI processes are employed, the number of coarse grid levels is 6, hybrid forward Gauss–Seidel is used as smoother, the overlap of RAS method is 0, and ILU(0) is used as the subdomain solver.

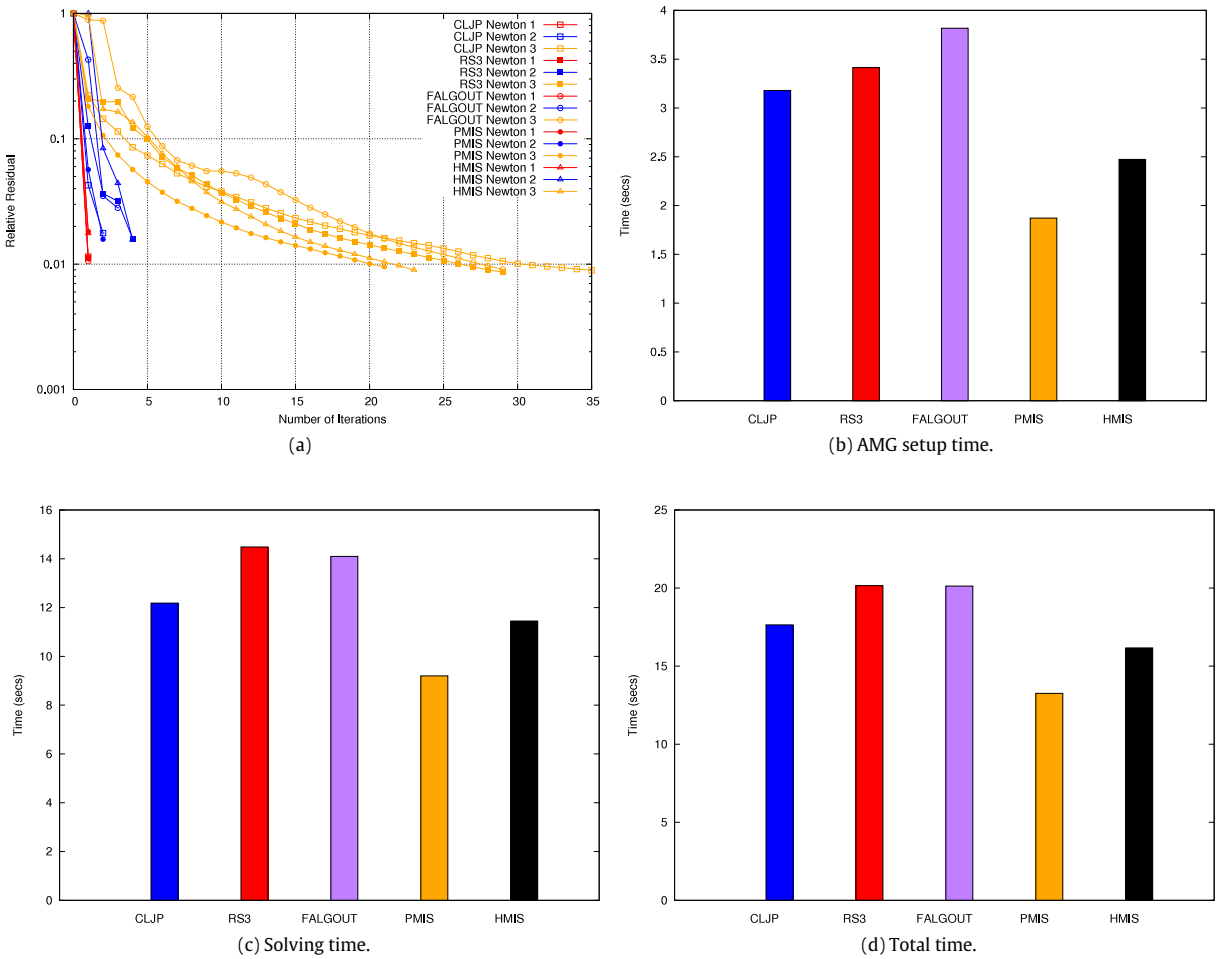
C.S.	# N	# L	A.L.N	AMG	L. Time
RS3	1051	33,370	31.8	1130	15,085
CLJP	1060	35,242	33.2	1176	15,769
FALGOUT	1072	35,873	33.5	1333	16,272
HMIS	**1079**	**31,658**	29.3	758	**13,911**
PMIS	**1125**	**33,319**	29.6	721	**14,392**

Table 4
Comparison of different smoothing method: SPE10 case. 144 MPI processes are employed, the number of coarse grid levels is 6, multipass interpolation is employed, the overlap of RAS method is 0, and ILU(0) is used as the subdomain solver.

C.S.	Smoother	T.S.	# N	# L	A.L.N	AMG	L. Time
	GS-F	**407**	**1417**	**55,080**	**38.9**	**191**	**3400**
	GS-B	**407**	**1415**	**52,358**	**37.0**	**184**	**3661**
PMIS	**GS-S**	**407**	**1411**	**51,781**	**36.7**	**181**	**3350**
	C–F Jacobi	407	1422	49,751	35.0	187	3663
	ℓ_1-Jacobi	407	1454	74,790	51.4	201	5068
	GS-F	**407**	**1415**	**51,474**	**36.4**	**229**	**3168**
	GS-B	**407**	**1423**	**52,428**	**36.8**	**229**	**3295**
HMIS	**GS-S**	**407**	**1415**	**50,544**	**35.7**	**217**	**3312**
	C–F Jacobi	407	1402	46,776	33.4	221	3533
	ℓ_1-Jacobi	407	1430	61,353	42.9	238	3971

submatrices, and "ILU" is the total time of factorization for submatrices. We can observe the similar phenomena: compared with the ILU(k) methods, the ILUT method with the above threshold parameters is slightly more expensive in the incomplete factorization process; however, it has much better performance not only on the number of nonlinear and linear iterations, but also on the computational time.

Besides the choice of a subdomain solver, the level of overlap also affects the efficiency of the RAS method. Usually, the more levels of overlap are used, the more accurate the solution for subdomain problems is, which should improve the convergence rate of the linear solver. However, it is interesting to see that the 0 level of overlap gives the best performance, the fewest nonlinear and linear iterations, and the least computational time; see Fig. 9(a)–(e). In Table 7, employing the ILUT



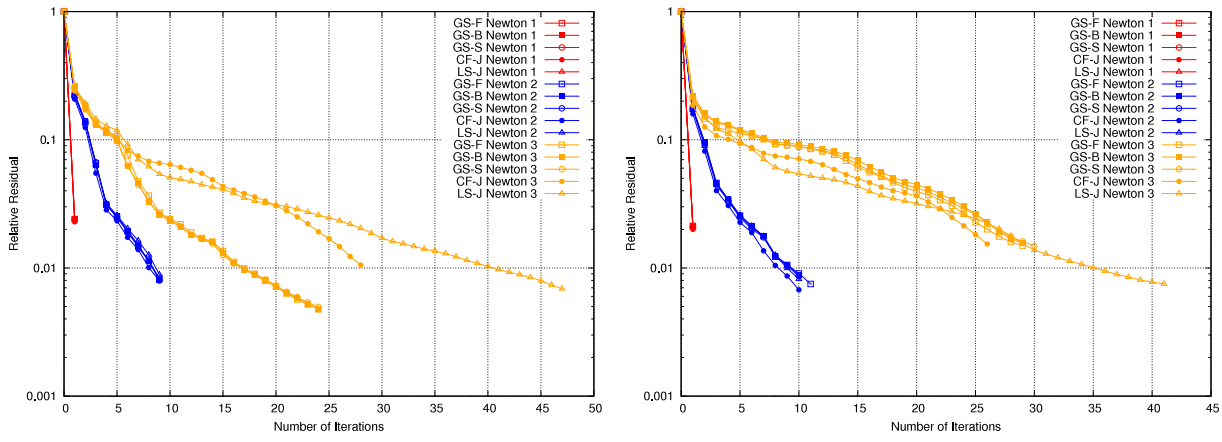
(a)

(b) AMG setup time.

(c) Solving time.

(d) Total time.

Fig. 3. Time cost of different coarsening strategies, refined SPE10 case.

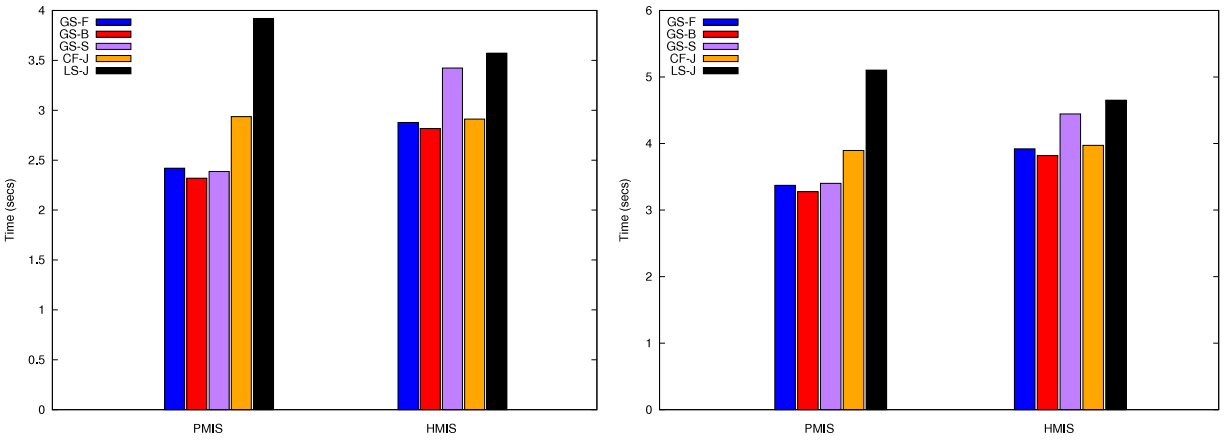


(a) PMIS coarsening.

(b) HMIS coarsening.

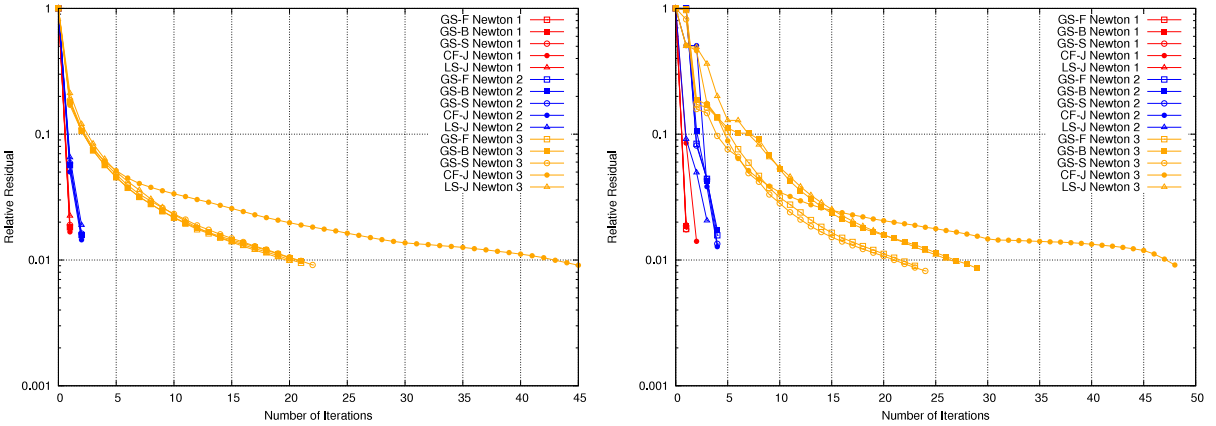
Fig. 4. Comparison of smoothers.

as the subdomain solver, we compare the overall performance of different overlap levels, and similar phenomena can be observed. Hence, we can conclude that 0 level of overlap is the optimal choice for large-scale parallel reservoir simulation.

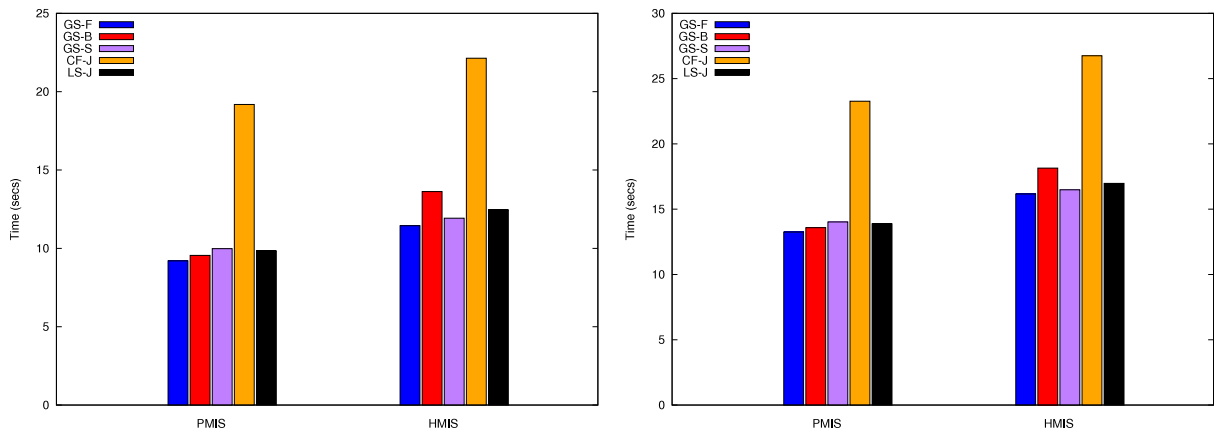


(a) Solving time. (b) Total time.

Fig. 5. Comparison of smoothers.



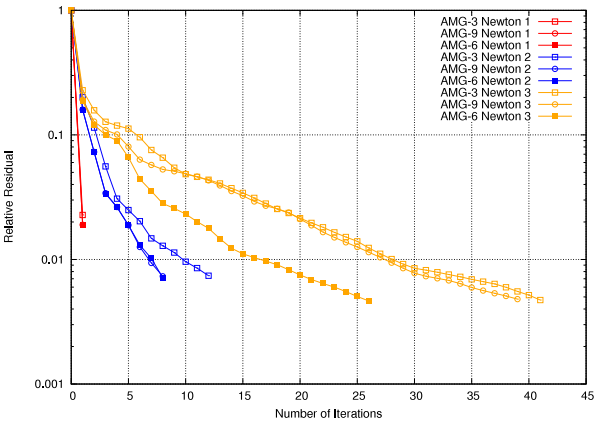
(a) PMIS coarsening. (b) HMIS coarsening.



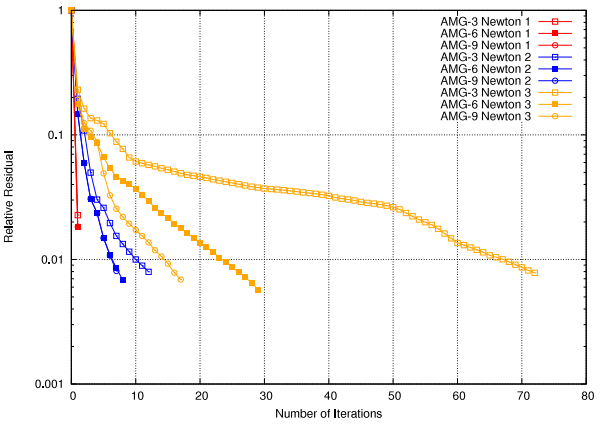
(c) Solving time. (d) Total time.

Fig. 6. Comparison of smoothers, refined SPE10 case.

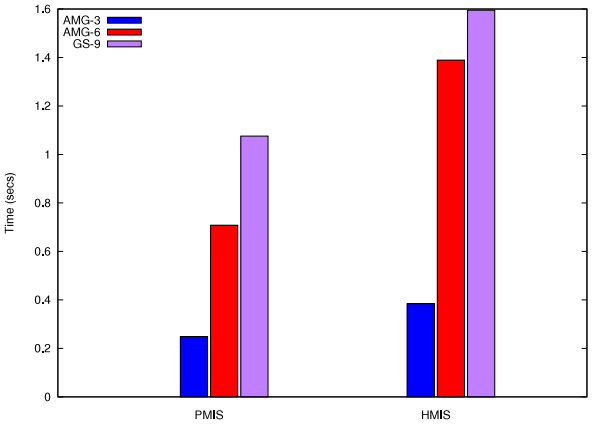
So far, we have compared the numerical results of different coarsening strategies, interpolations, smoothers and number of coarse grid levels in the AMG method, and different subdomain solvers and levels of overlap in the RAS method. From the



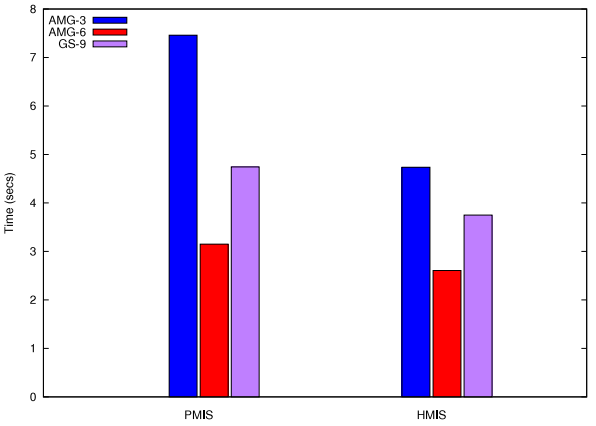
(a) PMIS coarsening strategy.



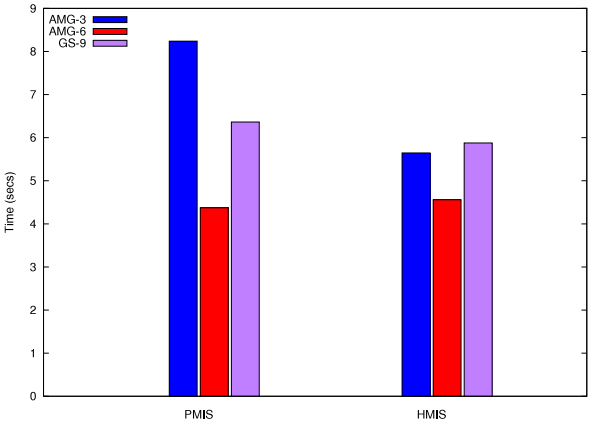
(b) HMIS coarsening strategy.



(c) AMG setup time.



(d) Solving time.



(e) Total time.

Fig. 7. Time cost of different coarse grid levels.

angle of computational efficiency, the HMIS and PMIS coarsening strategies combined with the multipass interpolation and forward (backward) hybrid Gauss–Seidel smoother in the AMG method and using the ILUT method as the subdomain solver with zero level of overlap in the RAS method show the best performance. At last, the scalability, which is as important as the efficiency, is tested and the results are shown in Table 8. Compared with the 12 processors cases, the scalability is satisfied for large scale parallel reservoir simulation. When 144 MPI processes are employed, there are only fewer than 10 thousands of grid cells, which leads to less perfect scalability.

Table 5
Comparison of different number of coarse grid levels, 144 MPI processes are employed, hybrid forward Gauss–Seidel is used as smoother, the overlap of RAS method is 0, multipass interpolation is employed, and ILU(0) is used as the subdomain solver.

C.S.	#C-Grid	T.S.	# N	# L	A.L.N	AMG	L. Time
PMIS	3	407	1384	72,635	52.5	100	3826
	6	**407**	**1417**	**55,080**	**38.9**	**191**	**3811**
	9	407	1507	46,913	31.1	398	5166
HMIS	3	407	1401	77,606	55.4	94	4021
	6	**407**	**1415**	**51,474**	**36.4**	**229**	**3570**
	9	407	1516	44,213	29.2	487	5175

Table 6
Comparison of different subdomain solver, 144 MPI processes are employed, the number of coarse grid levels is 6, multipass interpolation is employed, hybrid forward Gauss–Seidel is used as smoother, the overlap of RAS method is 0.

C.S.	Sub solver	T.S.	# N	# L	A.L.N	ASS	ILU	L. Time
PMIS	**ILUT**	**407**	**1404**	**37,338**	**26.6**	**15**	**81**	**2821**
	ILU(0)	407	1417	55,080	38.9	14	39	3871
	ILU(1)	407	1423	52,791	37.1	17	367	4314
	ILU(2)	407	1436	68,802	47.9	18	491	5159
HMIS	**ILUT**	**407**	**1415**	**37,198**	**26.3**	**16**	**85**	**2884**
	ILU(0)	407	1415	51,474	36.4	14	39	3570
	ILU(1)	407	1404	43,649	31.1	15	338	3376
	ILU(2)	407	1419	66,559	46.9	14	458	5665

Table 7
Comparison of overlap in RAS, 144 MPI processes are employed, the number of coarse grid levels is 6, multipass interpolation is employed, hybrid forward Gauss–Seidel is used as smoother and ILUT is used as the subdomain solver.

C.S.	Overlap	T.S.	# N	# L	A.L.N	ASS	ILU	L. Time
PMIS	**0**	**407**	**1404**	**37,338**	**26.6**	**15**	**81**	**2821**
	1	407	1431	71,154	49.7	1464	143	8256
	2	407	1441	53,839	37.4	641	161	7018
HMIS	**0**	**407**	**1415**	**37,198**	**26.3**	**16**	**85**	**2884**
	1	407	1433	77,811	54.3	286	140	6309
	2	407	1460	65,163	44.6	674	174	8129

Table 8
Scalability, the number of coarse grid levels is 6, multipass interpolation is employed, hybrid forward Gauss–Seidel is used as smoother, ILUT is used as the subdomain solver, and the overlap level is 0.

C.S.	NP	T.S.	# N	# L	A.L.N	L. Time	T. Time	Scalability
PMIS	12	407	1365	35,534	26.0	22,969	28,688	–
	36	407	1366	34,306	25.1	6,870	8,998	1.06
	72	407	1390	36,285	26.1	3,538	5,037	0.95
	144	407	1404	37,338	26.6	2,821	3,611	0.66
HMIS	12	407	1346	35,948	26.7	17,610	23,148	–
	36	407	1350	32,769	24.3	6,919	9,104	0.85
	72	407	1360	34,601	25.4	3,308	4,761	0.81
	144	407	1415	37,198	26.3	2,884	3,726	0.52

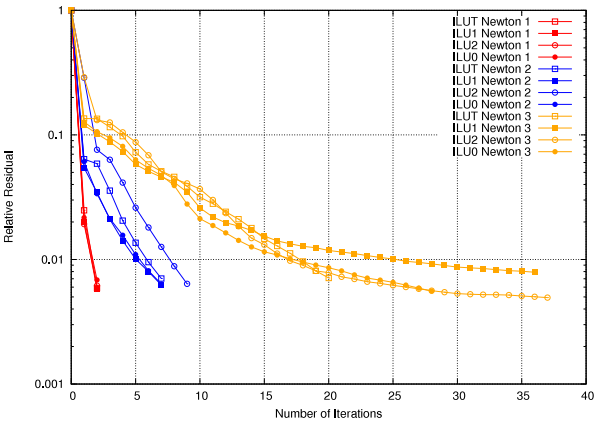
Table 9
Efficient settings of the CPR-type preconditioners.

AMG	Coarse Grid Levels	6
	Coarsening	HMIS/PMIS
	Interpolation	Multipass
	Smoother	Hybrid Gauss–Seidel
RAS	Overlap	0
	Subdomain solver	ILUT

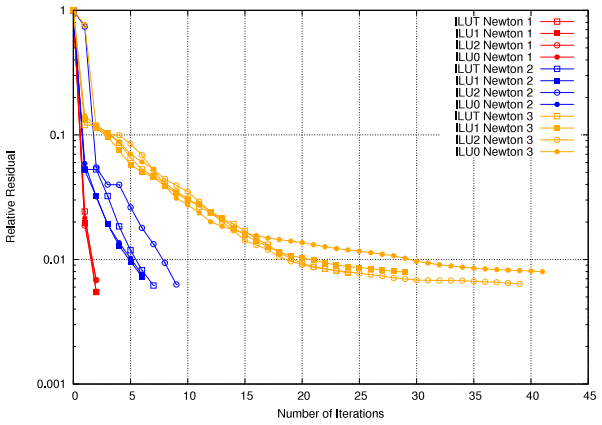
From the tests of this subsection, we can conclude the efficient settings of the CPR-type preconditioners in Table 9.

4.2. CPR-type preconditioners tests

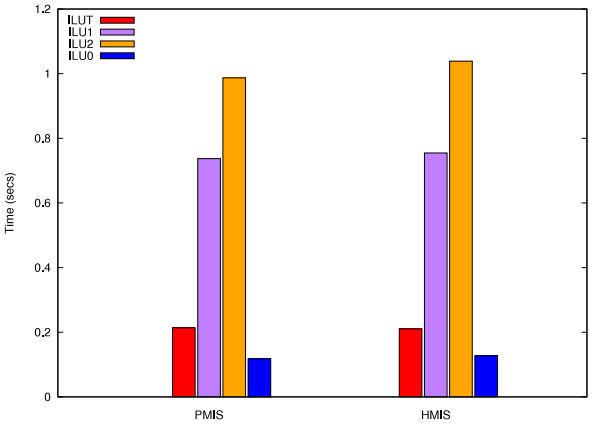
In this subsection, we will compare different CPR-type preconditioners, including the CPR, FASP, CPR-FPF and CPR-FFPF preconditioners. The efficient settings of the AMG and RAS methods, shown in Table 9, are employed by all the CPR-type preconditioners to test the cases in this subsection. The black oil SPE10 case and the refined case shown in 4.1



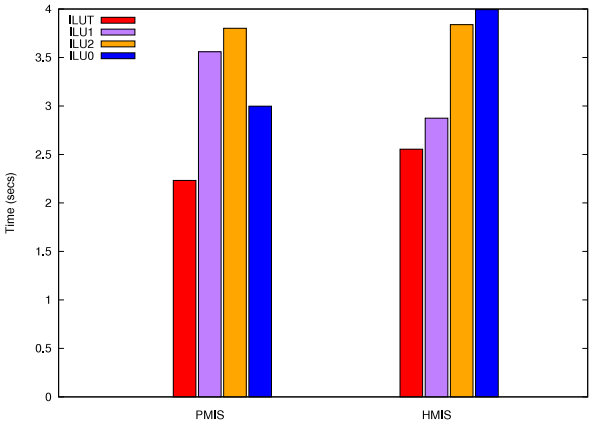
(a) PMIS coarsening.



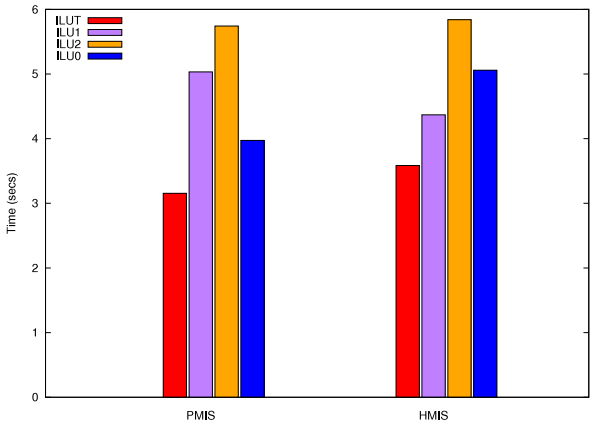
(b) HMIS coarsening.



(c) RAS setup time.



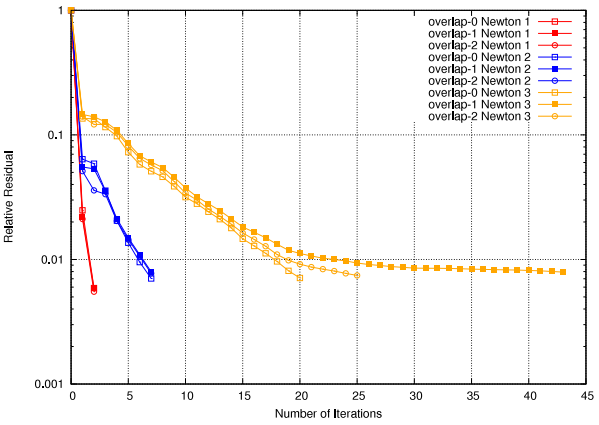
(d) Solving time.



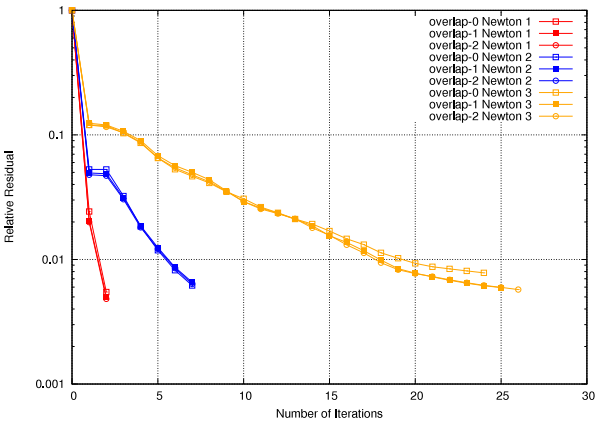
(e) Total time.

Fig. 8. Comparison of subdomain solvers in the RAS method.

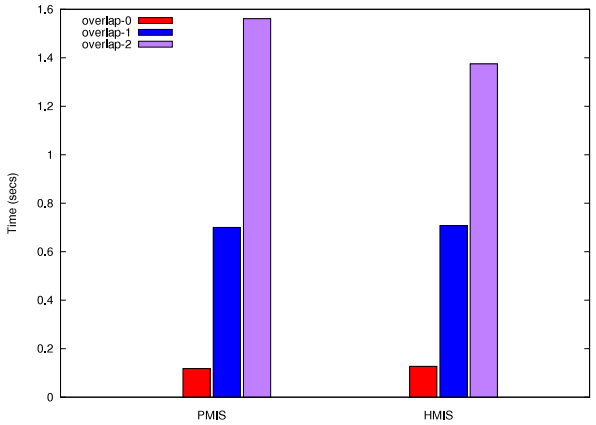
are tested. As we can see from Fig. 10a and b, the CPR-FPF and CPR-FFPF preconditioners are more effective than the other two preconditioners and employ fewer linear iterations. Compared with the CPR-FFPF preconditioner, the CPR-FPF preconditioner uses almost the same number of linear iterations, however it has less computational cost and uses less time; see Fig. 10e and f. The CPR preconditioner employs the most iterations, therefore it also uses more time than the CPR-FPF preconditioner even if it has less computational cost at each iteration. Compared with the CPR preconditioner, the FASP preconditioner has an additional stage to solve the saturation system R_S in Algorithm 2 in order to improve the efficiency.



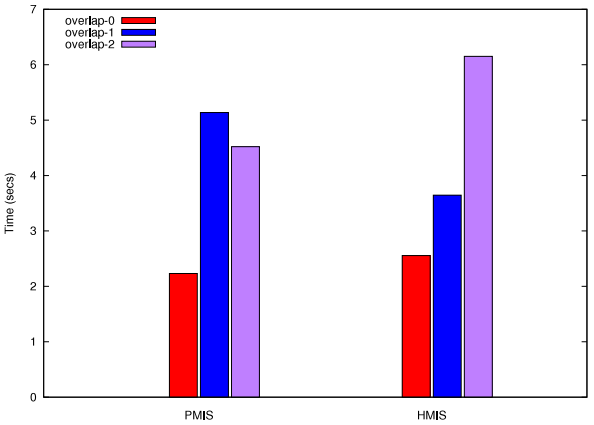
(a) PMIS coarsening.



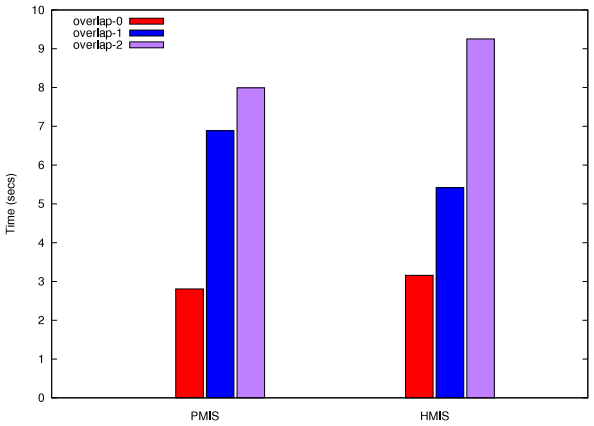
(b) HMIS coarsening.



(c) RAS setup time.



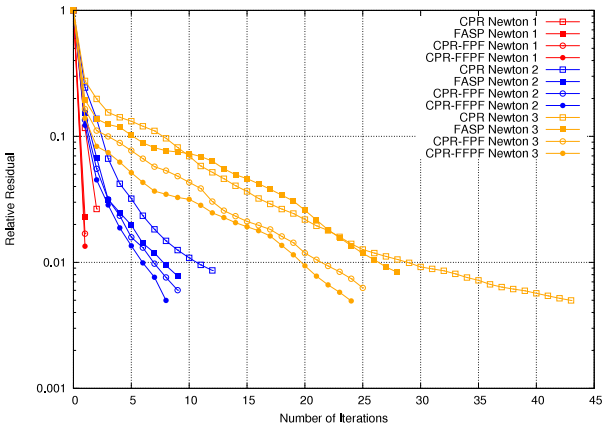
(d) Solving time.



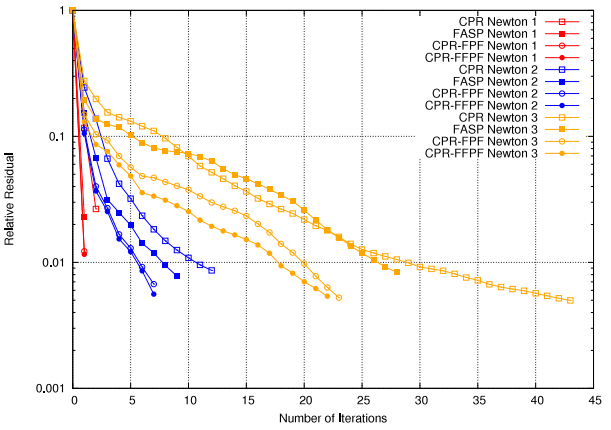
(e) Total time.

Fig. 9. Comparison of overlap levels in the RAS method.

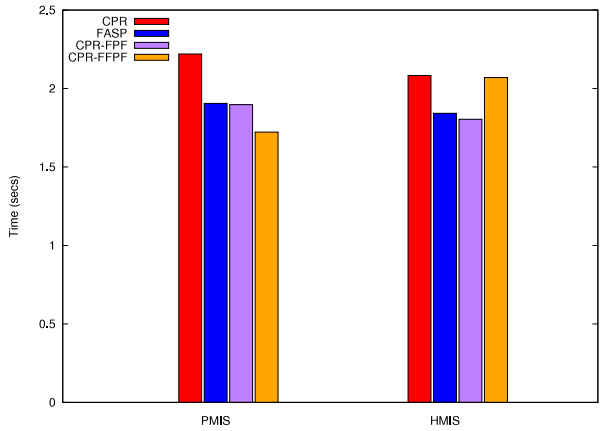
But the extra cost of the incomplete LU factorization for the saturation system and the solving processes makes the FASP preconditioner a more time consuming method than the CPR preconditioner. From Fig. 10a and b, we can see that the FASP preconditioner is less effective than the CPR-FPF preconditioner and leads to more linear iterations. Besides that, in the CPR-FPF and CPR-FFPF preconditioners, the saturation stage is replaced by a global smoother, so there is no need of extra setup time for the CPR-FPF preconditioner. Hence, the FASP preconditioner needs more time than the CPR-FPF preconditioner.



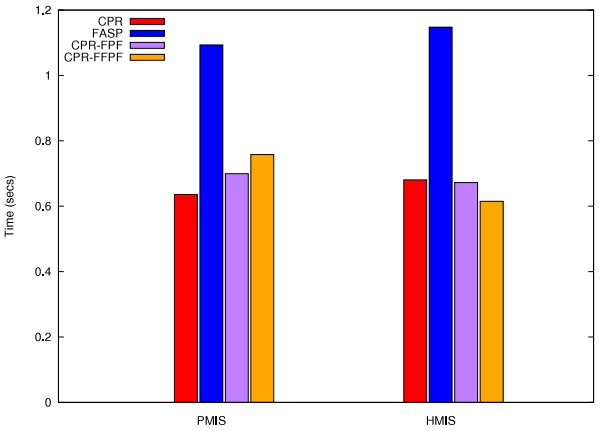
(a) PMIS coarsening.



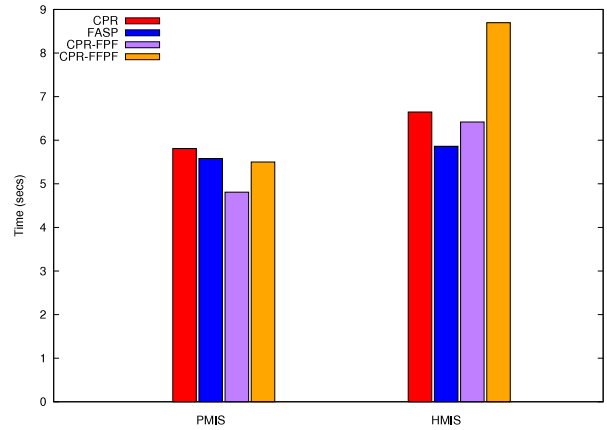
(b) HMIS coarsening.



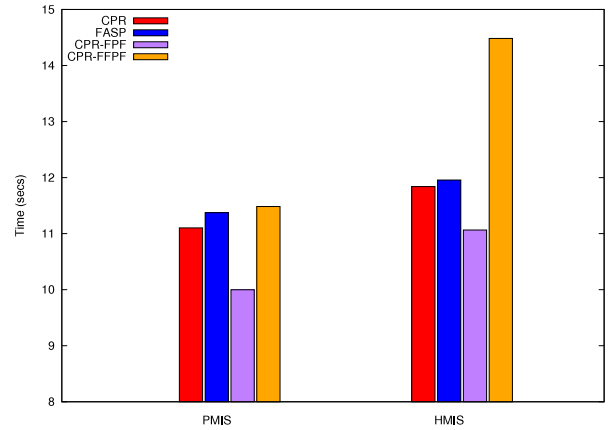
(c) AMG setup time.



(d) RAS setup time.



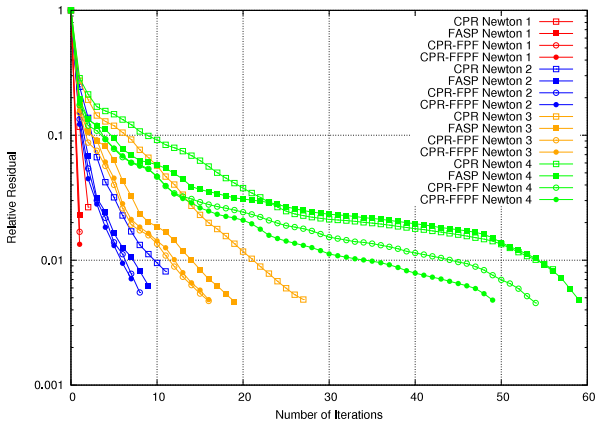
(e) Solving time.



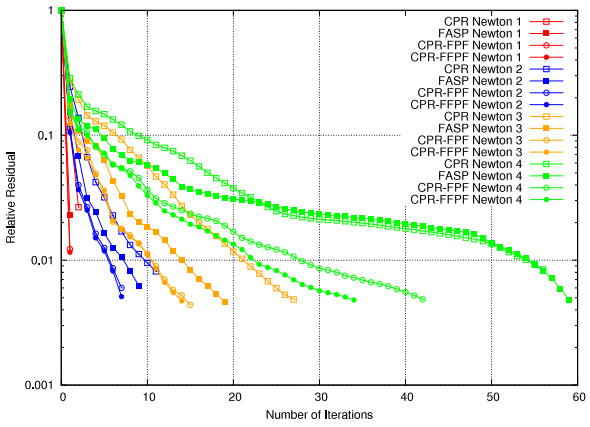
(f) Total time.

Fig. 10. Comparison of CPR-type preconditioners: SPE10. $\epsilon = 10^{-2}$.

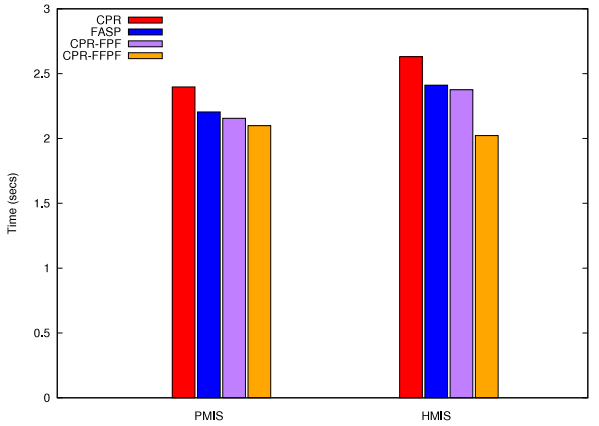
For reservoir simulation problems, smaller stopping criterion of Newton iterations is sometimes required to achieve more accurate prediction, which may lead to more nonlinear iterations and linear iterations. To further study the performance of the CPR-type preconditioners, we decrease the stopping criterion of Newton iterations ϵ in Eqs. (16)–(18) from 10^{-2} to 10^{-3}. From Fig. 11a and b, we can see that the CPR preconditioner is the least effective preconditioner for the third and



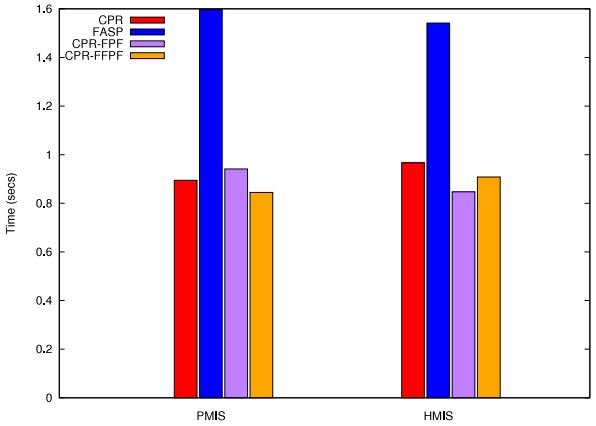
(a) PMIS coarsening.



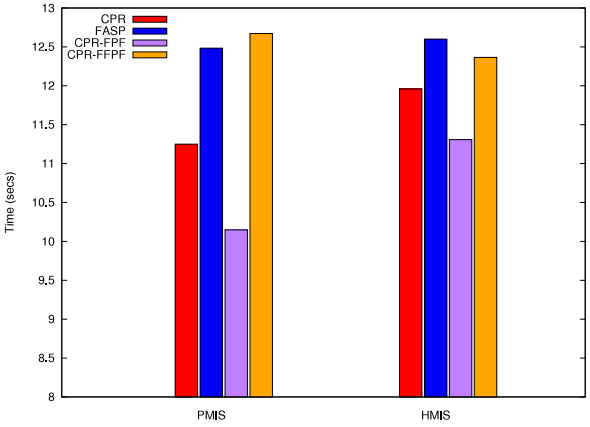
(b) HMIS coarsening.



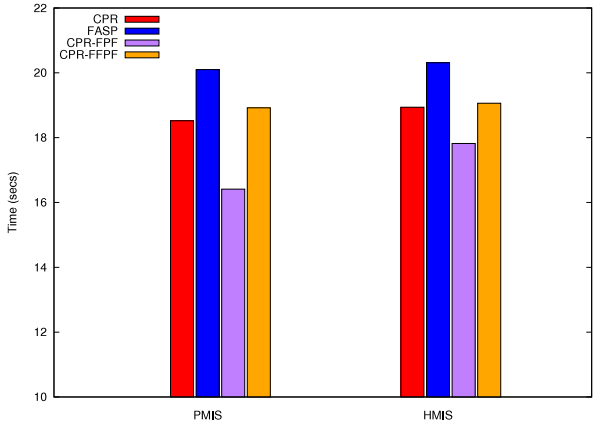
(c) AMG setup time.



(d) RAS setup time.



(e) Solving time.



(f) Total time.

Fig. 11. Comparison of CPR-type preconditioners: SPE10. $\epsilon = 10^{-3}$.

fourth nonlinear iterations and the FASP preconditioner has similar performance as the CPR preconditioner at the fourth nonlinear iteration. The CPR-FFPF and CPR-FPF preconditioners lead to less linear iterations and their performance is very close for the first three nonlinear iterations. The CPR-FFPF preconditioner employs less linear iterations than the CPR-FPF preconditioner at the fourth nonlinear iteration, however from the view of computational cost the CPR-FFPF is less efficient than the CPR-FPF preconditioner; see Fig. 11(e)–(f).

Table 10
Comparison of CPR-type preconditioners with different Newton stop criteria, SPE10.

ϵ	Preconditioner	T.S.	# N	# L	A.L.N	L. Time
10^{-2}	CPR	407	1415	37,198	26.3	2884
	FASP	407	1303	34,139	26.2	3987
	CPR-FPF	**407**	**1306**	**30,277**	**23.2**	**2867**
	CPR-FFPF	407	1298	31,763	24.5	3357
10^{-3}	CPR	407	1742	60,846	34.9	4810
	FASP	407	1574	53,988	34.3	6035
	CPR-FPF	**407**	**1571**	**45,719**	**29.1**	**4456**
	CPR-FFPF	407	1541	45,425	29.5	5063

Table 11
Comparison of CPR-type preconditioners with different Newton stop criteria, SPE10-refined, with the settings in Table 9.

ϵ	Solver	NP	# N	# L	A.L.N	L. Time	T. Time
10^{-2}	CPR	256	1045	26,845	25.7	30,140	37,595
		512	1040	27,538	26.5	17,131	21,258
		1024	1047	28,504	27.2	12,544	14,945
	FASP	256	1043	25,473	24.4	39,119	47,839
		512	1047	26,490	25.3	21,979	26,591
		1024	1049	26,797	25.5	15,213	17,888
	CPR-FPF	**256**	**1046**	**22,649**	**21.7**	**30,195**	**37,419**
		512	**1057**	**23,989**	**22.7**	**16,979**	**21,120**
		1024	**1059**	**24,503**	**23.1**	**12,327**	**14,742**
	CPR-FFPF	256	1034	21,926	21.2	33,176	40,283
		512	1040	23,028	22.1	18,308	22,434
		1024	1039	23,752	22.9	12,972	15,443
10^{-3}	CPR	256	1230	40,383	32.8	43,477	51,955
		512	1231	42,322	34.4	25,231	30,083
		1024	1238	43,480	35.1	18,421	21,232
	FASP	256	1237	39,237	31.7	56,985	67,071
		512	1235	41,232	33.4	32,997	38,598
		1024	1235	41,530	33.6	22,458	25,506
	CPR-FPF	**256**	**1232**	**32,866**	**26.7**	**42,081**	**50,619**
		512	**1230**	**34,339**	**27.9**	**23,675**	**28,512**
		1024	**1227**	**35,140**	**28.6**	**17,115**	**19,926**
	CPR-FFPF	256	1222	31,574	25.8	45,875	54,305
		512	1224	33,198	27.1	25,878	30,711
		1024	1226	35,332	28.8	18,583	21,407

In Table 10, the overall performance of the CPR-type preconditioners is summarized, from which we can see consistent phenomena as above. The CPR preconditioner performs the most linear iterations for both $\epsilon = 10^{-2}$ and $\epsilon = 10^{-3}$. However, it costs less time than the FASP and CPR-FFPF preconditioner, which benefits from its cheapest computational cost. Also because of its cheapest computational cost, the CPR preconditioner is less robust than other CPR-type preconditioners. When the nonlinear stop criterion ϵ is set to be smaller, 10^{-3}, the iterations of the CPR preconditioned linear solver increase, as well as the time cost. The CPR-FPF preconditioner is more effective than the FASP preconditioner and the CPR preconditioner, and it uses the least time when $\epsilon = 10^{-3}$. The CPR-FFPF preconditioner is the most robust and effective preconditioner, but the extra time cost of another global smoother makes it not worthwhile.

Because of the huge computational resource required and the limitation of the simulation time, 300 time steps are run for the refined SPE10 case. For all the four CPR-type preconditioners, considering the size of this case, from 256 to 1024 MPI processes are used. In Table 11, the similar phenomena are observed. Compared with the CPR and CPR-FPF preconditioners, the FASP preconditioner and the CPR-FFPF preconditioner cost more computational time. For this much larger case, the CPR-FPF preconditioner used the least time for both $\epsilon = 10^{-2}$ and $\epsilon = 10^{-3}$. Due to the less global smoothing, the computational time of the CPR preconditioner is close to that of the CPR-FPF preconditioner, even it has more linear iterations. This also benefits from the efficient settings in Table 9. Without these settings, the CPR-FPF will perform much better than the CPR preconditioner. For example, in Table 12, the RS3 coarsening strategy is used to replace the HMIS and PMIS coarsening strategies. Hence, the CPR-FPF preconditioner is the recommended preconditioner among all the CPR-type preconditioners.

5. Adaptive preconditioners

For the large-scale problems with highly heterogeneous permeability, the CPR-type preconditioners are more effective and efficient than the ILU preconditioners, However, due to the expensive computational cost of the CPR-type

Table 12
Comparison of CPR-type preconditioners with different Newton stop criteria, SPE10-refined, RS3 coarsening strategy.

ϵ	Solver	NP	# N	# L	A.L.N	L. Time	T. Time
10^{-3}	CPR	256	1237	41,270	33.3	51,365	60,866
		512	1236	42,388	34.2	27,204	32,197
		1024	1236	44,579	36.0	19,374	22,213
	FASP	256	1224	38,615	31.5	62,977	73,681
		512	1208	39,721	32.8	35,400	42,758
		1024	1208	41,499	34.3	23,308	26,380
	CPR-FPF	**256**	**1225**	**32,769**	**26.7**	**49,175**	**58,591**
		512	**1230**	**33,656**	**27.3**	**25,617**	**31,395**
		1024	**1210**	**34,001**	**28.1**	**17,180**	**19,960**
	CPR-FFPF	256	1214	32,515	26.7	52,663	61,162
		512	1214	34,093	28.0	30,858	35,993
		1024	1202	34,569	28.7	18,676	21,464

Table 13
Linear performance of a time step.

Nonlinear	Absolute residual	Relative residual	# L
1st	4.22×10^{-1}	3.56×10^{-2}	1
2nd	2.15×10^{-2}	6.63×10^{-3}	15
3rd	3.94×10^{-4}	1.11×10^{-2}	50
4th	4.93×10^{-6}	2.31×10^{-2}	73

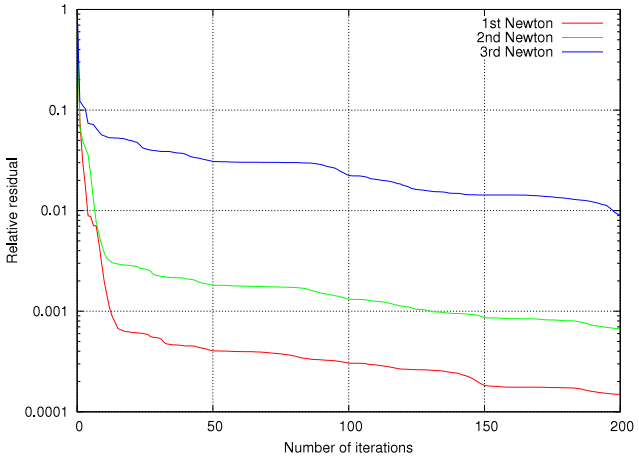


Fig. 12. Performance of RAS–ILU preconditioner, ILUT is used as the subdomain solver.

preconditioners, they may not be the most time-saving methods for all the reservoir simulation problems. For the cases relatively easier to solve, such as problems with homogeneous permeability, the ILU preconditioned linear solver may use less time, even if the iterations are much more than the CPR-type preconditioned linear solver. Moreover, since the inexact Newton method is used in the simulation, the linear iteration stop criterion depending on the residual of each nonlinear iteration may be around 0.1, which may be easy to achieve by ILU preconditioned linear solver. In this section, we aim to design an algorithm that can automatically choose the CPR-type preconditioners or ILU preconditioners according to each different problem. This idea is first mentioned in [36], and we will do more insight analysis and numerical experiments.

In Table 13, we randomly select a time step of the refined-SPE10 case and list the number of linear iterations, the initial absolute residual, the relative residual after linear iterations.

As we can see, when the initial absolute residual becomes smaller during the processes of nonlinear iterations, the linear system becomes more and more difficult to solve. For example, only one linear iteration is employed at the first nonlinear iteration, while there are 73 linear iterations to achieve the required accuracy at the fourth nonlinear iteration. The numerical behaviour shown in Table 13 is not a particular case, instead it can be observed from all the numerical experiments we present in this paper. Considering the high computational cost of CPR-type preconditioners, if using the RAS–ILU preconditioner to replace the CPR-type preconditioners for the linear systems that are relatively easy to solve, the computational time may be saved. We first examine the performance of the RAS–ILU preconditioner. The RAS–ILU preconditioner is used to solve a randomly selected time step on its own; see Fig. 12. A common phenomenon can be observed for each Newton iteration, which is the high frequency errors can be removed very quickly through the first several

iterations while the convergence becomes really difficult when the relative residual reaches a specific value. If the required relative residual of a linear system is big enough then the RAS–ILU preconditioned linear solver is capable of effectively handling it, the huge computational cost of AMG setup and solving could be saved. When the RAS–ILU preconditioner is not effective and convergence rate of linear iteration is close to 1, it is natural to switch the RAS–ILU preconditioner to a CPR-type preconditioner. For example, if we expect the RAS–ILU preconditioned linear solver to converge in $\mathcal{N}_e$ times of iterations, an expected convergence rate R_e can be calculated by

$$R_e = \epsilon^{\frac{1}{\mathcal{N}_e}}, \tag{21}$$

where ϵ is the stop criterion of the linear iterations. The relative residual of the linear systems e is monitored during the iterations. Once its convergence rate is larger than R_e, the RAS–ILU preconditioner is considered as ineffective and will be switched to CPR-type preconditioners. Another obvious phenomenon from Fig. 12 is that the RAS–ILU preconditioner becomes less effective during the processes of Newton iterations. It means that the linear systems become more difficult along the Newton iterations, and in this situation more effective preconditioner should be used. Hence, we introduce an indicator $\varepsilon_{a,r}^{n,k}$ to decide the difficult level of solving a linear system

$$A^{n,k}x = b^{n,k}, \tag{22}$$

which resulted from the kth nonlinear iteration of nth time step. The initial absolute residual $\varepsilon_a^{n,1}$ of the linear system resulted from the first nonlinear iteration is used as a benchmark, the indicator can be defined as

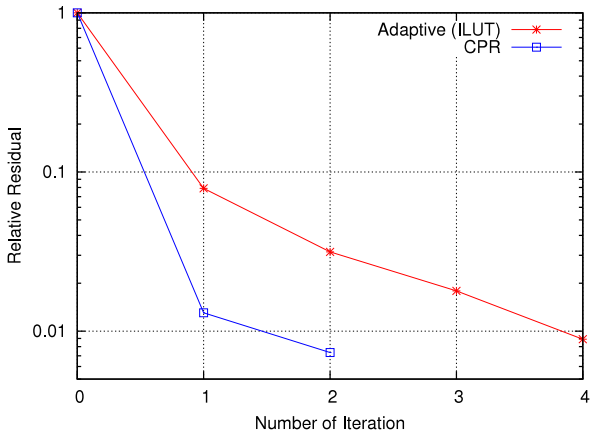
$$\varepsilon_{a,r}^{n,k} = \frac{\varepsilon_a^{n,k}}{\varepsilon_a^{n,1}}, \tag{23}$$

where $\varepsilon_a^{n,1}$ is the initial absolute residual of linear system (22). This indicator can be seen as a relatively initial absolute residual. If $\varepsilon_{a,r}$ is larger than a switch criterion ε_s, we assume the linear system is relatively easy to solve, and as a sequence the RAS–ILU preconditioner is used; otherwise, the CPR-type preconditioners should be used. If the switch criterion $\varepsilon_s = 1.0$ or $\varepsilon_s = 0.0$, the adaptive preconditioner will degenerate to the CPR-type preconditioners or the RAS–ILU preconditioner, respectively. The smaller the switch criterion is set, the adaptive strategy is considered as more aggressive. In our computation, the switch criterion ε_s is set to 10^{-3}. Full description of this adaptive preconditioner strategy is written in Algorithm 5. Restarted GMRES(m) method and the CPR-FPF preconditioner are used as representative linear solver and the CPR-type preconditioner in Algorithm 5, and any other linear solver and CPR-type preconditioner can be used as an alternate.

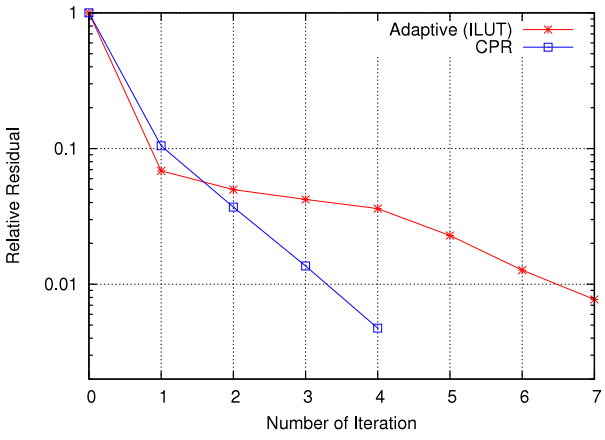
Algorithm 5 Adaptive Linear Solver: ALS(ϵ, $\mathcal{N}_e$, $\varepsilon_a^{n,0}$, ε_s)

1: Calculate the expected convergence rate $R_e = \epsilon^{\frac{1}{\mathcal{N}_e}}$.
2: Calculate the initial absolute residual $\varepsilon_a^{n,k} = \|A^{n,k}x - b^{n,k}\|$ and the indicator $\varepsilon_{a,r}^{n,k} = \frac{\varepsilon_a^{n,k}}{\varepsilon_a^{n,0}}$.
3: **if** $\varepsilon_{a,r}^{n,k} > \varepsilon_s$ **then**
4: Setup RAS–ILU preconditioner and employ RAS–ILU preconditioned GMRES iterations.
5: **while** the relative tolerance $e > \epsilon$ **do**
6: calculate the convergence rate $R_{\text{ILU}} = e^{\frac{1}{\mathcal{N}_{\text{ILU}}}}$, where $\mathcal{N}_{\text{ILU}}$ is the number of RAS–ILU preconditioned GMRES iterations.
7: **if** $R_{\text{ILU}} > R_e$ **then**
8: **goto** (14).
9: **else**
10: **continue** the RAS–ILU preconditioned GMRES iterations.
11: **end if**
12: **end while**
13: **else**
14: Setup the CPR-FPF preconditioner and employ CPR-FPF preconditioned GMRES iterations.
15: **while** the relative tolerance $e > \epsilon$ **do**
16: **continue** the CPR-FPF preconditioned GMRES iterations.
17: **end while**
18: **end if**

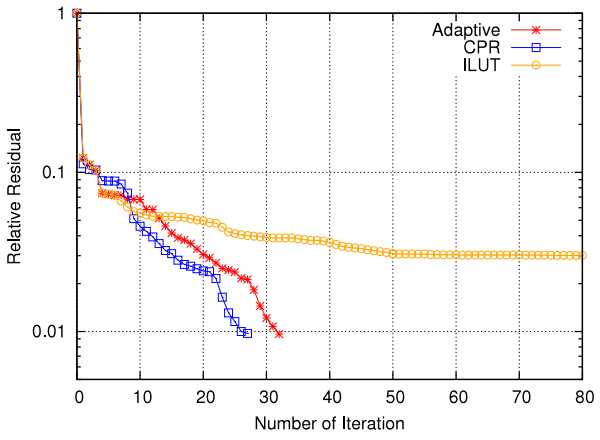
We test this idea in three Newton iterations of one time step and the results are shown in Fig. 13(a)–(c). The relative residual is set to 0.01 for each Newton iteration. For the first and second Newton iterations, the RAS–ILU preconditioned linear solver is capable of achieving the required stop criteria. For the last Newton iteration, the RAS–ILU preconditioner is switched to a CPR-type preconditioner after several iterations. From Fig. 13(d), we can see that the total linear solver time has been significantly reduced for the first two Newton iterations since the AMG setup time and AMG V-cycle time have been saved and the computational cost of RAS–ILU preconditioner is lower.



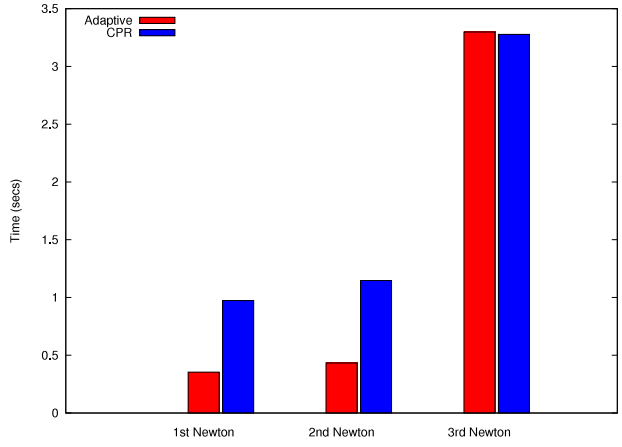
(a) 1st Newton iteration.



(b) 2nd Newton iteration.



(c) 3rd Newton iteration.



(d) Linear solver time.

Fig. 13. Comparison of the RAS–ILU, CPR and adaptive preconditioners.

To further illustrate the efficiency of the adaptive preconditioning strategy, the black oil SPE10 case, the refined case and a black oil case with hundreds of wells (MX521 case) are tested. The MX521 case contains $512 \times 469 \times 20 = 4.88$ million grid cells, as well as 90 water injection wells and 110 production wells. The porosity and permeability are both constant for this case. The simulation time is 2000 days. More details about this case can be found in [37]. The switch criterion ε_s is set to be 10^{-3}, and the expected iteration times $\mathcal{N}_e$ is 80. The efficient setting in Table 9 is used in the numerical experiments of this subsection. HMIS is used as the coarsening strategy, and hybrid forward Gauss–Seidel is used as smoother.

For the black oil SPE10 case, the CPR preconditioner and the CPR-FPF preconditioner, which give the most efficient performance for $\epsilon = 10^{-2}$ and $\epsilon = 10^{-3}$, are compared with the adaptive preconditioner by using 36, 72 and 144 MPI processes, Tables 14 and 15. When the nonlinear stop criterion ϵ is set to be 10^{-2}, see Table 14, the adaptive solver saves more than a half of the CPR-type preconditioning processes and employs the cheaper preconditioner, RAS–ILU, to replace them. After increasing the accuracy of nonlinear iterations ϵ to 10^{-3}, more accurate solutions of the linear iterations are required, then the adaptive preconditioning strategy employs more CPR preconditioning processes to keep its effectiveness, see Table 15. Consequently, the linear solver time and the total simulation time both decrease as expected. Moreover, the total number of linear iterations also decreases, which exceeds our expectations.

For the refined black oil SPE10 case, a comparison between the CPR-FPF preconditioner and the adaptive preconditioner is given in Tables 16 and 17. For this much larger case, from 256 to 1024 MPI processes are used. Similar phenomena are observed for both $\epsilon = 10^{-2}$ and $\epsilon = 10^{-3}$. In Table 16, around half of the CPR-FPF preconditioners are replaced by RAS–ILU preconditioners. When $\epsilon = 10^{-3}$, since more accurate solutions are required, smaller percentage of CPR-FPF preconditioners should be replaced, which is also captured by the adaptive strategy; see Table 17. The adaptive strategy brings in more linear iterations than the CPR-FPF preconditioned linear solver. However, the computational cost of linear iterations decreases, and about one fourth of the computational time is saved.

Table 14
Performance of the adaptive preconditioning strategy, SPE10. The nonlinear stop criterion ϵ is 10^{-2}.

Solver	NP	T.S.	# N	# L	ILU/CPR	A.L.N	L. Time	T. Time
CPR	36	407	1308	35,307	–	27.0	7424	9663
	72	407	1318	39,439	–	29.9	4382	5926
	144	407	1321	42,045	–	31.8	2708	4404
Adaptive	36	407	1352	34,695	21,635/13,060	25.6	5171	7426
	72	407	1375	36,198	21,980/14,218	26.3	2749	4358
	144	407	1386	38,054	23,278/14,776	27.4	1847	3546

Table 15
Performance of the adaptive preconditioning strategy, SPE10. The nonlinear stop criterion ϵ is 10^{-3}.

Solver	NP	T.S.	# N	# L	ILU/CPR	A.L.N	L. Time	T. Time
CPR-FPF	36	407	1542	44,995	–	29.1	11,182	13,812
	72	407	1545	47,693	–	30.8	5,965	7,766
	144	407	1545	55,119	–	35.6	4,314	6,223
Adaptive	36	407	1735	43,369	20,332/23,037	24.9	7,656	10,469
	72	407	1760	45,255	20,852/24,403	25.7	3,983	5,990
	144	407	1789	47,548	21,827/25,721	26.5	2,639	4,763

Table 16
Performance of the adaptive preconditioning strategy, refined-SPE10. The nonlinear stop criterion ϵ is 10^{-2}.

Solver	NP	# N	# L	ILU/CPR	A.L.N	L. Time	T. Time
CPR-FPF	256	1046	22,649	–	21.7	30,195	37,419
	512	1057	23,989	–	22.7	16,979	21,120
	1024	1059	24,503	–	23.1	12,327	14,742
Adaptive	256	1026	30,245	20,115/10,130	29.5	23,121	30,254
	512	1038	30,587	20,112/10,475	29.5	11,585	15,706
	1024	1062	32,455	20,756/11,699	30.6	8,702	11,226

Table 17
Performance of the adaptive preconditioning strategy, refined-SPE10. The nonlinear stop criterion ϵ is 10^{-3}.

Solver	NP	# N	# L	ILU/CPR	A.L.N	L. Time	T. Time
CPR-FPF	256	1232	32,866	–	26.7	42,081	50,619
	512	1230	34,339	–	27.9	23,675	28,512
	1024	1227	35,140	–	28.6	17,115	19,926
Adaptive	256	1232	38,707	20,146/18,561	31.4	33,412	41,901
	512	1232	39,444	20,444/19,000	32.0	17,836	22,840
	1024	1225	40,220	20,798/19,442	32.8	12,239	15,168

The linear systems that resulted from the MX521 case are relatively easy to solve due to the constant permeability. Hence, the RAS–ILU preconditioner (ILUT is used as the subdomain solver) is also effective enough for this case. The results of these three types of preconditioners are compared in Table 18. We can see that the CPR-FPF preconditioned linear solvers use the longest computational time although they employ the fewest nonlinear and linear iterations. Benefiting from the low computational cost, the RAS–ILU preconditioned linear solvers are more efficient than the CPR-FPF preconditioned ones. For this easier case, the adaptive preconditioning strategy combines both advantages of the RAS–ILU and CPR-FPF preconditioners. A big portion of RAS–ILU preconditioners is employed by the adaptive strategy to save the computational time, and a small portion of the CPR-FPF preconditioners is used to keep the preconditioning processes highly effective. Consequently, the linear and nonlinear iterations are less than the RAS–ILU preconditioners, the computational time is the least among the three types of preconditioners. We need to mention that the efficient settings in Table 9 of AMG and RAS methods are another important factors that make the adaptive preconditioner more efficient than the RAS–ILU preconditioner for this case. If the efficient settings are not used, the CPR-FPF preconditioner will cost some extra computational time, which makes the RAS–ILU the most efficient preconditioner.

6. Conclusions

In this paper, we have briefly reviewed the CPR-type preconditioners, which are recognized as the most efficient preconditioners in reservoir simulation. As two of its most important parts, the AMG method and the RAS method have also been reviewed. In parallel reservoir simulation, there are a lot of optional components in the AMG and RAS methods, which affect the efficiency of the CPR-type preconditioners. In order to find the efficient settings, taking the SPE10 case and the refined SPE10 case as examples, we have systematically compared these components in detail. According to the numerical results, the combination of HMIS (or PMIS), multipass interpolation, hybrid Gauss–Seidel smoother, six levels of

Table 18
Performance of the adaptive preconditioning strategy, MX521 case. The nonlinear stop criterion ϵ is 10^{-3}.

Solver	NP	# N	# L	ILU/CPR	A.L.N	L. Time	T. Time
CPR-FPF	128	5148	11,912	–	2.3	14,488	27,451
	256	5210	12,248	–	2.4	8,335	15,437
	512	5297	12,627	–	2.4	3,996	7,433
	1024	5363	12,897	–	2.4	2,172	4,049
RAS–ILU	128	7053	41,739	–	5.9	10,584	24,466
	256	7037	43,624	–	6.2	5,862	13,274
	512	6945	45,817	–	6.6	2,908	6,623
	1024	6882	48,924	–	7.1	1,396	3,413
Adaptive	128	5790	34,175	31,359/2816	5.9	10,396	23,455
	256	5627	35,167	32,211/2956	6.2	5,943	12,000
	512	5764	36,694	33,707/2987	6.4	2,949	6,437
	1024	5774	38,992	35,804/3188	6.8	1,417	3,234

coarse grids, ILUT subdomain solver and zero level of overlap, is the most efficient and shows satisfying scalability. With these efficient settings, different CPR-type preconditioners are compared, and we conclude that the CPR-FPF preconditioner is more robust for the extremely large-scale problems and the problems requiring high accuracy. In order to improve the efficiency of the CPR-type preconditioners, an adaptive strategy is designed. The adaptive strategy successfully use the RAS–ILU preconditioner to solve the relatively easier linear systems and use the CPR-type preconditioners to solve the rest, and saves the computational time.

Acknowledgements

The support of Department of Chemical and Petroleum Engineering, University of Calgary and Reservoir Simulation Research Group is gratefully acknowledged. The research is partly supported by NSERC/AIEES/Foundation CMG, AITF iCore, IBM Thomas J. Watson Research Center, and the Frank and Sarah Meyer FCMG Collaboration Centre for Visualization and Simulation. The research is also enabled in part by support provided by WestGrid (www.westgrid.ca), SciNet (www.scinethpc.ca) and Compute Canada Calcul Canada (www.computecanada.ca).

References

[1] Z. Chen, G. Huan, Y. Ma, Computational Methods for Multiphase Flows in Porous Media, Vol. 2, Siam, 2006.
[2] K. Stüben, A review of algebraic multigrid, J. Comput. Appl. Math. 128 (1) (2001) 281–309.
[3] K. Stüben, An introduction to algebraic multigrid, multigrid, Multigrid (2001) 413–532.
[4] J.W. Ruge, K. Stüben, Algebraic multigrid, Multigrid Methods 3 (1987) 73–130.
[5] K. Stüben, Algebraic multigrid (AMG): an introduction with applications, in: GMD-Forschungszentrum Informationstechnik, 1999.
[6] U.M. Yang, Boomeramg: a parallel algebraic multigrid solver and preconditioner, Appl. Numer. Math. 41 (1) (2002) 155–177.
[7] A. Cleary, R. Falgout, V. Henson, J. Jones, Coarse-grid selection for parallel algebraic multigrid, in: Solving Irregularly Structured Problems in Parallel, Springer Berlin Heidelberg, 1998, pp. 104–115.
[8] H.D. Sterck, U. Yang, J. Heys, Reducing complexity in parallel algebraic multigrid preconditioners, SIAM J. Matrix Anal. Appl. 27 (4) (2006) 1019–1039.
[9] H.D. Sterck, R.D. Falgout, J.W. Nolting, U.M. Yang, Distancetwo interpolation for parallel algebraic multigrid, Numer. Linear Algebra Appl. 15 (2–3) (2008) 115–139.
[10] U.M. Yang, On longrange interpolation operators for aggressive coarsening, Numer. Linear Algebra Appl. 17 (2–3) (2010) 453–472.
[11] M. Adams, M. Brezina, J. Hu, R. Tuminaro, Parallel multigrid smoothing: polynomial versus GaussSeidel, J. Comput. Phys. 182 (2) (2003) 593–610.
[12] A.H. Baker, R.D. Falgout, T.V. Kolev, U.M. Yang, Multigrid smoothers for ultraparallel computing, SIAM J. Sci. Comput. 33 (5) (2011) 2864–2887.
[13] A. Elli, O.B. Widlund, Domain Decomposition Methods: Algorithms and Theory (Vol. 3), Springer, Berlin, 2005.
[14] D. Maksymilian, O.B. Widlund, Domain decomposition algorithms with small overlap, SIAM J. Sci. Comput. 15 (3) (1994) 604–620.
[15] T. Chan, T. Mathew, Domain decomposition algorithms, Acta Numer. 3 (1994) 61–143.
[16] Bjorstad, Petter, W. Gropp, Domain Decomposition: Parallel Multilevel Methods for Elliptic Partial Differential Equations, Cambridge university press, 2004.
[17] X.-C. Cai, M. Sarkis, A restricted additive Schwarz preconditioner for general sparse linear systems, SIAM J. Sci. Comput. 21 (2) (1999) 792–797.
[18] J.R. Wallis, Incomplete Gaussian elimination as a preconditioning for generalized conjugate gradient acceleration, in: SPE Reservoir Simulation Symposium, 1983.
[19] G.A. Behie, P.A.F. Jr., Incomplete factorization methods for fully implicit simulation of enhanced oil recovery, SIAM J. Sci. Stat. Comput. 5 (3) (1984) 541–561.
[20] J.R. Wallis, R.P. Kendall, T.E. Little, Constrained residual acceleration of conjugate residual methods, in: SPE Reservoir Simulation Symposium, 1985.
[21] H. Cao, H.A. Tchelepi, J.R. Wallis, H.E. Yardumian, Parallel scalable unstructured CPR-type linear solver for reservoir simulation, in: SPE Annual Technical Conference and Exhibition, 2005.
[22] T.M. Al-Shaalan, H.M. Klie, A.H. Dogru, M.F. Wheeler, Studies of robust two stage preconditioners for the solution of fully implicit multiphase flow problems, in: SPE Reservoir Simulation Symposium, 2009.
[23] H. Liu, K. Wang, Z. Chen, A family of constrained pressure residual preconditioners for parallel reservoir simulations, Numer. Linear Algebra Appl. (2015) 1–32.
[24] H. Liu, K. Wang, Z. Chen, K.E. Jordan, Efficient multi-stage preconditioners for highly heterogeneous reservoir simulations on parallel distributed systems, in: SPE Reservoir Simulation Symposium. Society of Petroleum Engineers, 2015.
[25] K. Wang, H. Liu, Z. Chen, A scalable parallel black oil simulator on distributed memory parallel computers, J. Comput. Phys. 301 (2015) 19–34.

[26] X. Hu, W. Liu, G. Qin, J. Xu, C. Zhang, Development of a fast auxiliary subspace pre-conditioner for numerical reservoir simulators, in: SPE Reservoir Characterisation and Simulation Conference and Exhibition, 2011.
[27] H. Liu, K. Wang, Z. Chen, K.E. Jordan, H. Deng, J. Luo, A parallel framework for reservoir simulators on distributed-memory supercomputers, in: SPE/IATMI Asia Pacific Oil & Gas Conference and Exhibition, 2015.
[28] R.D. Falgout, U.M. Yang, Hypre: A library of high performance preconditioners, in: Computational Science-ICCS 2002, Springer Berlin Heidelberg, 2002, pp. 632–641.
[29] Z. Chen, M. Espedal, R.E. Ewing, Continuous-time finite element analysis of multiphase flow in groundwater hydrology, Appl. Math. 40 (3) (1995) 203–226.
[30] S. Lacroix, Y.V. Vassilevski, M.F. Wheeler, Decoupling preconditioners in the implicit parallel accurate reservoir simulator (IPARS), Numer. Linear Algebra Appl. 8 (8) (2001) 537–549.
[31] R. Scheichl, R. Masson, J. Wendebourg, Decoupling and block preconditioning for sedimentary basin simulations, Comput. Geosci. 7 (2003) 295–318.
[32] R.E. Bank, T.F. Chan, W.M.C. Jr., R.K. Smith, The Alternate-Block-Factorization procedure for systems of partial differential equations, BIT Numer. Math. 29 (4) (1989) 938–954.
[33] Z. Chen, D.J. Jim, Approximation of coefficients in hybrid and mixed methods for nonlinear parabolic problems, Mat. Apl. Comput. 10 (1) (1991) 137–160.
[34] T. Chen, N. Gewecke, Z. Li, A. Rubiano, R. Shuttleworth, B. Yang, X. Zhong, Fast computational methods for reservoir flow models, 2009.
[35] M.A. Christie, M.J. Blunt, Tenth SPE comparative solution project: A comparison of upscaling techniques., SPE Reser. Eval. Eng. 4 (4) (2001) 308–317 4.4 (2001): 308-317.
[36] K. Wang, H. Liu, J. Luo, Z. Chen, A multi-continuum multi-phase parallel simulator for large-scale conventional and unconventional reservoirs, J. Natur. Gas Sci. Eng. 33 (2016) 483–496.
[37] G.L. Brown, D.A. Collins, Z. Chen, et al., Efficient preconditioning for algebraic multigrid and red-black ordering in adaptive-implicit black-oil simulations, in: SPE Reservoir Simulation Symposium, Society of Petroleum Engineers, 2015.