CCF Transactions on High Performance Computing (2023) 5:144–159
https://doi.org/10.1007/s42514-023-00136-0

REGULAR PAPER

An improved multistage preconditioner on GPUs for compositional reservoir simulation

Li Zhao[1] · Shizhe Li[2] · Chen-Song Zhang[2] · Chunsheng Feng[1,3,4] · Shi Shu[1,3]

Received: 11 August 2022 / Accepted: 5 January 2023 / Published online: 1 February 2023

Abstract
The compositional model is often used to describe multicomponent multiphase porous media flows in the petroleum industry. The fully implicit method with strong stability and weak constraints on time-step sizes is commonly used in mainstream commercial reservoir simulators. In this paper, we develop an efficient multistage preconditioner for the fully implicit compositional flow simulation. The method employs an adaptive setup phase to improve the parallel efficiency on GPUs. Furthermore, a multicolor Gauss–Seidel algorithm based on the adjacency matrix is applied in the algebraic multigrid methods for the pressure part. Numerical results demonstrate that the proposed algorithm achieves good parallel speedup while yielding the same convergence behavior as the corresponding sequential version.

Keywords Compositional model · Fully implicit method · multistage preconditioner · multicolor Gauss–Seidel · GPU · Compute unified device architecture (CUDA)

AMS Classification 49M20 · 65F10 · 68W10 · 76S05

1 Introduction

The compositional model, which allows the fluids to be composed of various material components, is a widely-used mathematical model for describing multiphase flows in porous media (Aziz and Settari 1979; Chen et al. 2006). The compositional model is an extension of the black oil model (Peaceman 1977), which is formed by multiple coupled nonlinear partial differential equations. Some complicated oil displacement technologies can be accurately simulated based on the compositional model, such as polymer flooding, surfactant and alkali oil displacement agents, and miscible flooding.

Numerical methods for compositional numerical simulation are abundant; to name a few, IMplicit Pressure Explicit Concentrations (IMPEC) method (Fussell and Fussell 1979), Fully Implicit Method (FIM) (Coats 1980), IMplicit Pressure/SATuration and explicit concentrations (IMPSAT) (Quandalle and Savary 1989), and Adaptive Implicit Method (AIM) (Collins et al. 1992). In the IMPEC method, pressure is implicit and other variables are explicit. One of its advantages is that no need to solve coupled linear algebra systems, but its time stepsize is constrained by the Courant–Friedrichs–Lewy (CFL) (Courant et al. 1928) condition. The

✉ Li Zhao
zhaoli@smail.xtu.edu.cn

Shizhe Li
lishizhe@lsec.cc.ac.cn

Chen-Song Zhang
zhangcs@lsec.cc.ac.cn

Chunsheng Feng
spring@xtu.edu.cn

Shi Shu
shushi@xtu.edu.cn

[1] School of Mathematics and Computational Science, Xiangtan University, Xiangtan 411105, People's Republic of China

[2] LSEC & NCMIS, Academy of Mathematics and Systems Science, Chinese Academy of Sciences, and School of Mathematical Sciences, University of Chinese Academy of Sciences, Beijing 100190, People's Republic of China

[3] Hunan Key Laboratory for Computation and Simulation in Science and Engineering, Xiangtan University, Xiangtan 411105, People's Republic of China

[4] National Center for Applied Mathematics in Hunan, Hunan Shaofeng Institute for Applied Mathematics, Xiangtan 411105, People's Republic of China

FIM method, on the contrary, is unconditionally stable concerning time stepsizes because all variables are handled implicitly. The IMPSAT method is a combination of the IMPEC and FIM methods, where pressure and saturation variables are handled implicitly and the molar fractions are calculated explicitly. The AIM method is also a compromise between the IMPEC and FIM methods and it also yields Jacobian algebraic systems that are easier to solve than FIM.

The FIM method possesses the characteristics of good stability and is widely used in commercial simulators. However, a coupled nonlinear system of equations needs to be solved at each time step of FIM. Such systems are usually linearized using the Newton-type methods, which require solving a coupled Jacobian linear algebra system during each iteration. In the numerical simulation, the solution of these systems is the main computational cost (Zhang 2022). Therefore, efficient linear solvers are crucial for improving the efficiency of fully implicit reservoir simulation, especially for large-scale three-dimensional problems.

The linear solution methods generally consist of a setup phase (SETUP) and a solve phase (SOLVE). Iterative methods are widely used in petroleum reservoir simulation due to their low memory overhead and good parallel scalability. More specifically, the GMRES and BiCGstab methods (Saad 2003) are exploited to solve the nonsymmetric systems that arise from the fully implicit discretization of reservoir models. The preconditioning techniques are crucial to speeding up the convergence of iterative methods (Zhang 2022). For large-scale reservoir simulation, multistage preconditioners are very competitive. The classical Constrained Pressure Residual (CPR) (Cao et al. 2005; Li et al. 2017; Wallis 1983; Wallis et al. 1985) approach is a well-known two-stage preconditioner. It utilizes the Algebraic MultiGrid (AMG) (Brandt et al. 1984; Falgout 2006) method to approximate the inverse of the pressure matrix in the first stage and the Incomplete LU (ILU) factorization (Meyerink 1983) to smooth the overall reservoir matrix in the second stage. The MultiStage Preconditioner (MSP) (Al-Shaalan et al. 2009; Hu et al. 2013; Stüben et al. 2007) is a generalization of the CPR method, which is also widely used in petroleum reservoir problems.

Parallel computing is an important approach to improving the speed of simulation and a lot of attention has been paid to developing efficient parallel algorithms. A Graphics Processing Unit (GPU) with thousands of cores is a parallel accelerator that is designed to handle images and graphics originally. Due to its high float-point performance and memory bandwidth (NVIDIA 2022), it has great potential in petroleum reservoir simulation. In recent years, research on GPU parallel algorithms has been developed in (Sudan et al. 2010; Chen et al. 2014; Yang et al. 2016; Kang et al. 2018; Manea and Almani 2019; Middya et al. 2021; Esler et al. 2022) and the references therein. For example, (Chen et al. 2014; Yang et al. 2016) developed a hybrid sparse matrix storage format and the corresponding sparse matrix-vector multiplication (SpMV) kernel. Kang et al. (2018) developed a parallel nonlinear solver based on OpenACC (OpenACC 2022) using the domain decomposition method to achieve load balancing. Finally, Manea and Almani (2019) studied a parallel algebraic multiscale solver on GPU architectures to improve the solution efficiency of the pressure equation.

In this paper, we focus on a GPU-based parallel linear solver for compositional models, which is an extension of the recent work for the black oil model (Zhao et al. 2022). Such an extension is mainly different in the following three aspects. Firstly, the model and choice of primary variables are different. For the black oil model, the primary variables are oil pressure, water saturation, and oil saturation, while for the compositional model, the primary variables are reference pressure and overall molar concentration of components. Potentially they could give rise to different algebra systems. Secondly, the two models calculate the physical parameters of the fluid differently. The black oil model gets the fluid properties by looking up (or interpolating) the tabular data of fluid properties in terms of pressure, while the compositional model solves the equation of state for them (Peng and Robinson 1977; Chen et al. 2006). Finally, there are a lot of numerical studies on the algebraic solvers for the fully implicit discretization of the black oil model (Cao et al. 2005; Li et al. 2017; Wallis 1983; Wallis et al. 1985; Brandt et al. 1984; Falgout 2006; Meyerink 1983; Al-Shaalan et al. 2009; Hu et al. 2013; Stüben et al. 2007). On the other hand, to the best of our knowledge, only a few numerical tests have been done for the compositional model in the literature. Therefore, we wish to study the numerical performance of the proposed solver for the compositional model and compare it with the widely-used solvers in commercial software. Moreover, in this work, we developed an improved parallel multistage preconditioning method for the compositional model for GPUs. The main contributions of this work are listed as follows:

- We propose a multistage preconditioner with an adaptive SETUP procedure, denoted as ASMSP. The proposed method can significantly reduce the number of SETUP calls, so as to reduce the computational overhead and improve parallel efficiency.
- We investigate a multicolor Gauss–Seidel (GS) algorithm based on algebraic grouping for the smoothing operator in the AMG methods. This algorithm has been shown to produce same convergence behavior as the corresponding sequential algorithm on multicore CPUs (Zhao et al. 2022).

- The proposed methods are integrated into the open-source simulator OpenCAEPoro (OpenCAEPoro 2022) for multicomponent multiphase flow in porous media.

The rest of the paper is organized as follows. Sect. 2 briefly introduces the compositional model and its fully implicit discretization. In Sect. 3, an MSP preconditioner with adaptive SETUP is developed. In Sect. 4, the parallel implementation of multicolor GS based on the adjacency matrix is proposed. In Sect. 5, numerical experiments are performed to evaluate the convergence and parallel speedup of the proposed method. Section 6 summarizes the work of this paper.

2 Mathematical model and discretization

2.1 The compositional model

In this paper, we consider the isothermal multicomponent compositional model (Aziz and Settari 1979; Chen et al. 2006) containing n_c components (hydrocarbon and water) and n_p phases (including at least the water phase). The mass conservation equation for the component i reads

$$\frac{\partial}{\partial t}\left(\phi \sum_{j=1}^{n_p} x_{ij}\xi_j S_j\right) + \nabla\cdot\left(\sum_{j=1}^{n_p} x_{ij}\xi_j \boldsymbol{u}_j\right) = Q_i,\ i = 1,2,\dots,n_c, \tag{1}$$

where ϕ is the porosity of the rock, x_{ij} is the molar fraction (dimensionless) of component i in phase j, ξ_j is the molar concentration of phase j, S_j is the saturation of phase j, $\boldsymbol{u}_j$ is the velocity of phase j, and Q_i is source/sink terms of component i.

Assume that the pore volume of the porous media is filled with the fluid, the volume balance equation is then

$$V = V_{\text{pore}} := \phi V_{\text{bulk}}, \tag{2}$$

where V is the fluid volume, V_{pore} is the pore volume, and V_{bulk} is the bulk volume.

Assume that the phase j fluid in porous media satisfies the Darcy's law:

$$\boldsymbol{u}_j = -\frac{\kappa\kappa_{rj}}{\mu_j}\left(\nabla P_j - \rho_j \mathfrak{g}\nabla z\right),\ j = 1,2,\dots,n_p, \tag{3}$$

where κ is the absolute permeability, κ_{rj} is the relative permeability of phase j, μ_j is the viscosity coefficient of phase j, P_j is the pressure of phase j, ρ_j is the density of phase j, $\mathfrak{g}$ is the gravity acceleration, and z is the depth.

Moreover, the variables S_j, x_{ij} and P_j in the Eqs. (1)–(3) satisfy the following constitutive relations:

- Saturation constraint equation:

$$\sum_{j=1}^{n_p} S_j = 1. \tag{4}$$

- Molar fraction constraint equation:

$$\sum_{i=1}^{n_c} x_{ij} = 1,\ j = 1,2,\dots,n_p. \tag{5}$$

- Capillary pressure equation:

$$P_j = P - P_{cj}(S_j),\ j = 1,2,\dots,n_p, \tag{6}$$

where P is the reference pressure, and $P_{cj}(S_j)$ is the capillary pressure between the reference phase and phase j, which will be ignored in the rest of this paper.

2.2 Discretization method

In this section, we first simplify the compositional model, then describe the choice of main equations and primary variables, and finally present the FIM discretization method.

2.2.1 The choices of primary variables

To begin with, we introduce the overall molar concentration N_i and molar flux $\boldsymbol{F}_i$ of the component i, which are defined as

$$N_i = \phi \sum_{j=1}^{n_p} x_{ij}\xi_j S_j, \tag{7}$$

$$\boldsymbol{F}_i = \sum_{j=1}^{n_p} x_{ij}\xi_j \boldsymbol{u}_j. \tag{8}$$

Equation (1) can be simplified to

$$\frac{\partial}{\partial t}N_i + \nabla\cdot\boldsymbol{F}_i = Q_i,\ i = 1,2,\dots,n_c. \tag{9}$$

In this paper, we choose the mass conservation Eq. (9) and the volume balance Eq. (2) as the main equations, and there are $n_c + 1$ equations in total. The reference pressure P and the overall molar concentration N_i $(i = 1,2,\dots,n_c)$ are used as the primary variables of the discrete method, and there are $n_c + 1$ variables in total. After solving the primary variables, the fluid volume state function $V(P, N_1,\dots,N_{n_c})$ can be obtained by the equation of state and flash calculation, see (Peng and Robinson 1977; Chen et al. 2006) for more details.

2.2.2 Finite volume method and backward Euler method

The finite volume method (FVM) (LeVeque 2002) features simplicity, conservation, adaptivity to complex geometric regions, and monotonicity; it is a commonly used discretization method in the petroleum industry. In this paper, the spatial domain is discretized by using FVM. Suppose that the spatial domain $\Omega \subset \mathbb{R}^3$ (an open set) is discretized into m elements set $\{\tau_k\}_{k=1}^m$ (the shape of the element is not considered here), which satisfies $\cup_{k=1}^m \overline{\tau}_k = \overline{\Omega}$ and $\tau_k \cap \tau_\ell = \emptyset,\ k \neq \ell,\ k, \ell = 1, \dots, m$.

$$\int_{\tau_k} \frac{\partial}{\partial t} N_i dV + \int_{\tau_k} \nabla \cdot \boldsymbol{F}_i dV = \int_{\tau_k} Q_i dV,\ i = 1, 2, \dots, n_c, \tag{10}$$

using the divergence theorem, we can get

$$\int_{\tau_k} \frac{\partial}{\partial t} N_i dV + \int_{S_k} \boldsymbol{F}_i \cdot \boldsymbol{n} dS = \int_{\tau_k} Q_i dV,\ i = 1, 2, \dots, n_c, \tag{11}$$

where $S_k := \partial \tau_k$ is all the surfaces set of element τ_k and $\boldsymbol{n}$ is the outer unit normal vector to S_k.

Therefore, the discrete equation on element τ_k can be written as

$$\frac{\partial}{\partial t} N_{i,k} + \sum_{s \in S_k} \boldsymbol{F}_{i,s} = Q_{i,k},\ i = 1, 2, \dots, n_c, \tag{12}$$

where the discrete flux $\boldsymbol{F}_{i,s}$ can be defined in various ways (e.g., Aavatsmark 2002; Aavatsmark et al. 2008), we consider the following form

$$\boldsymbol{F}_{i,s} = \left\{ \frac{L\kappa}{d} \right\}_s \sum_{j=1}^{n_p} \left(\left\{ x_{ij} \xi_j \frac{\kappa_{rj}}{\mu_j} \right\}_{s,\text{up}} \delta_s (P + P_{cj} - \rho_j \mathfrak{g} z) \right). \tag{13}$$

here, L and d denote the size of the interface and the distance between two adjacent elements, respectively. δ_s denotes the difference between the values on the two adjacent elements. Because the primary variables of the discrete equations are defined at the center of element, the harmonic mean value and the upstream weighted value are usually used to approximate the value $\{\cdot\}_s$ and $\{\cdot\}_{s,\text{up}}$ of the physical quantities on the interface s (Chen et al. 2006; Zhang 2022).

For the semi-discrete Eq. (12), the time derivative term is discretized by using the backward Euler method, and the superscripts n and $n+1$ denote the time t_n and t_{n+1}, respectively. The fully discrete volume balance equation and mass conservation equations are

$$V^{n+1} - V^{n+1}_{\text{pore}} = 0, \tag{14}$$

$$\frac{N^{n+1}_{i,k} - N^n_{i,k}}{\Delta t} + \sum_{s \in S_k} \boldsymbol{F}^\star_{i,s} = Q^\star_{i,k},\ i = 1, 2, \dots, n_c, \tag{15}$$

respectively. Here the source/sink term is simplified into a known function; but in practical problems, it is related to the production mode of the well in the oil field, and is also strongly coupled with the primary variables. Since the focus of this paper is not how to handle the well equations, we do not describe it in detail.

Note that the fully discrete Eqs. (14) and (15) are nonlinear, and the terms $\boldsymbol{F}^\star_{i,s}$ and $Q^\star_{i,k}$ are subject to be specified. Below, we will give their expressions.

2.2.3 Fully implicit method

The FIM scheme is currently commonly used in mainstream commercial reservoir simulators. This is because the scheme has the characteristics of strong stability and weak constraint on the timestep sizes. These characteristics highlight the advantages of the FIM, especially when the nonlinearity of the models is relatively strong.

When both $\boldsymbol{F}^\star_{i,s}$ and $Q^\star_{i,k}$ in Eq. (15) take the value of t_{n+1} time, the fully implicit discrete equations are

$$\frac{N^{n+1}_{i,k} - N^n_{i,k}}{\Delta t} + \sum_{s \in S_k} \boldsymbol{F}^{n+1}_{i,s} = Q^{n+1}_{i,k},\ i = 1, 2, \dots, n_c. \tag{16}$$

Owing to the implicit solution for $n_c + 1$ primary variables, Eqs. (14) and (16) are strongly coupled nonlinear systems of equations that need to be linearized. In this work, we exploit the well-known Newton's method to linearize Eqs. (14) and (16). The Jacobian equation for increments $\delta P, \delta N_1, \dots, \delta N_{n_c}$ can be written as

$$\frac{1}{\Delta t} \alpha_P \delta P - \frac{1}{\Delta t} \sum_{i=1}^{n_c} \alpha_i \delta N_i = r_P, \tag{17}$$

$$\begin{aligned} &\frac{1}{\Delta t} \delta N_i - \nabla \cdot (\beta_i \nabla \delta P) - \nabla \cdot (\beta_{iP} \nabla \delta P) \\ &\quad - \sum_{k=1}^{n_c} \nabla \cdot (\beta_{ik} \nabla \delta N_k) = r_i,\ i = 1, \dots, n_c, \end{aligned} \tag{18}$$

where the coefficients α_P, α_i, β_i, β_{iP}, and β_{ik} are obtained by partial derivation of the model coefficients with respect to P or N_i; see (Qiao 2015) for details.

2.2.4 Discrete system

After discretization, the coupled nonlinear algebraic equations are obtained. Such equations are linearized by adopting the

Newton method to form the sparse Jacobian system $Ax = b$ of the reservoir equation with implicit wells, namely:

$$\begin{pmatrix} A_{RR} & A_{RW} \\ A_{WR} & A_{WW} \end{pmatrix} \begin{pmatrix} x_R \\ x_W \end{pmatrix} = \begin{pmatrix} b_R \\ b_W \end{pmatrix}, \tag{19}$$

where A_{RR} and A_{RW} are the derivatives of the reservoir equations for reservoir variables and well variables, respectively; A_{WR} and A_{WW} are the derivatives of the well equations for reservoir variables and well variables, respectively; x_R and x_W are reservoir and bottom-hole flowing pressure variables, respectively; and b_R and b_W are the right-hand side vectors that correspond to the reservoir fields and the implicit wells, respectively.

The subsystem corresponding to the reservoir equations in the discrete system (19) is $A_{RR}x_R = b_R$; that is,

$$\begin{pmatrix} A_{PP} & A_{PN_1} & A_{PN_2} & \cdots & A_{PN_{n_c}} \\ A_{N_1P} & A_{N_1N_1} & A_{N_1N_2} & \cdots & A_{N_1N_{n_c}} \\ A_{N_2P} & A_{N_2N_1} & A_{N_2N_2} & \cdots & A_{N_2N_{n_c}} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ A_{N_{n_c}P} & A_{N_{n_c}N_1} & A_{N_{n_c}N_2} & \cdots & A_{N_{n_c}N_{n_c}} \end{pmatrix} \begin{pmatrix} x_P \\ x_1 \\ x_2 \\ \vdots \\ x_{n_c} \end{pmatrix} = \begin{pmatrix} b_P \\ b_1 \\ b_2 \\ \vdots \\ b_{n_c} \end{pmatrix}, \tag{20}$$

where P is reference pressure and N_i $(i = 1, 2, \ldots, n_c)$ are the overall molar concentration.

Remark 1 Our simulator used the classical Peaceman model (Peaceman 1977) for the well equations. For convenience, we omit the details of the well equations and refer readers to Peaceman (1977), Aziz and Settari (1979), Chen et al. (2006).

3 Parallel multistage preconditioners with adaptive SETUP

In the compositional model, the primary variables consist of reference pressure P and overall molar concentration N_i $(i = 1, 2, \ldots, n_c)$. These variables possess different mathematical properties, such as the parabolicity of the pressure equation and hyperbolicity of the concentration equations. These properties provide a theoretical basis for the construction of multiplicative subspace correction methods (Xu 1992, 1996); see Zhang (2022) for a recent review.

3.1 Multistage preconditioner

We first define two transfer operators of the reservoir matrix, suppose $\Pi_N : \mathcal{V}_N \to \mathcal{V}$ and $\Pi_P : \mathcal{V}_P \to \mathcal{V}$, where $\mathcal{V}_N$ and $\mathcal{V}_P$ are the overall molar concentration and pressure variables space, respectively, and $\mathcal{V}$ is the variables space of the whole reservoir. Then, the multiplicative multistage preconditioner B (Al-Shaalan et al. 2009; Hu et al. 2013; Stüben et al. 2007) is defined as

$$I - BA = (I - RA)(I - \Pi_P B_P \Pi_P^T A)(I - \Pi_N B_N \Pi_N^T A), \tag{21}$$

where the relaxation operator R employs the Block ILU (BILU) method, B_P and B_N are solved by the AMG and Block GS (BGS) methods, respectively.

Suppose that the mathematical behavior of preconditioner B acting on a known vector g is

$$w = Bg. \tag{22}$$

The corresponding multistage preconditioning algorithm (Feng et al. 2014) is shown in the Algorithm 3.1.

```
Algorithm 3.1 MSP preconditioning method
Require: $A, g, w, \Pi_N, \Pi_P$;
Ensure: $w = Bg$.
1: $r = g - Aw$;
2: $w = w + \Pi_N B_N \Pi_N^T r$;
3: $r = g - Aw$;
4: $w = w + \Pi_P B_P \Pi_P^T r$;
5: $r = g - Aw$;
6: $w = w + Rr$.
```

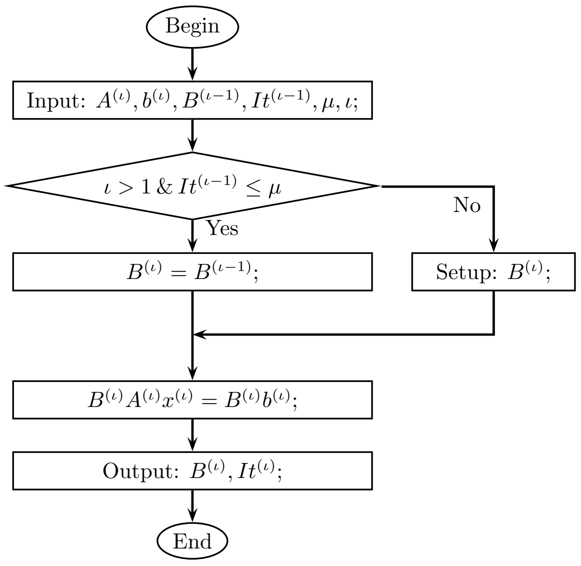

Begin
Input: $A^{(\iota)}, b^{(\iota)}, B^{(\iota-1)}, It^{(\iota-1)}, \mu, \iota$;
$\iota > 1 \,\&\, It^{(\iota-1)} \leq \mu$
Yes
No
$B^{(\iota)} = B^{(\iota-1)}$;
Setup: $B^{(\iota)}$;
$B^{(\iota)}A^{(\iota)}x^{(\iota)} = B^{(\iota)}b^{(\iota)}$;
Output: $B^{(\iota)}, It^{(\iota)}$;
End

Fig. 1 The algorithm flow chart of adaptive SETUP

3.2 MSP with adaptive SETUP

In this subsection, we utilize an adaptive SETUP strategy for the MSP preconditioner to improve its parallel efficiency, denoted as ASMSP. As mentioned earlier, this strategy has been employed for the black oil model by Zhao et al. (2022). Assume that the solution objective is $A^{(\iota)}x^{(\iota)} = b^{(\iota)}$. It is worth mentioning that superscript ι is the number of Newton iterations. This is because the reservoir model is a nonlinear system of equations, which is linearized using Newton's method here (see Chen et al. (2006) for more details). Figure 1 presents the algorithm flow chart of adaptive SETUP.

In Fig. 1, the main difference from the standard methods is that the preconditioner $B^{(\iota)}$ is yielded by an adaptive strategy. This strategy can be divided into the following two cases:

(1) The preconditioner $B^{(\iota)}$ inherits the information from the previous preconditioner $B^{(\iota-1)}$. A natural approach is to use the number of iterations $It^{(\iota-1)}$, required by solving the previous Jacobian system $A^{(\iota-1)}x^{(\iota-1)} = b^{(\iota-1)}$. We introduce a threshold μ (a non-negative integer); if $It^{(\iota-1)} \leq \mu$, the previous preconditioner $B^{(\iota-1)}$ is used as the preconditioner $B^{(\iota)}$.

(2) The preconditioner $B^{(\iota)}$ is regenerated; if $\iota = 1$ or $It^{(\iota-1)} > \mu$, the preconditioner $B^{(\iota)}$ is generated by calling Algorithm 3.1.

Remark 2 If the sizes of $A^{(\iota-1)}$ and $A^{(\iota)}$ are not the same, the preconditioner $B^{(\iota)}$ must be regenerated for sure.

Below, we illustrate the rationale for this approach. The number of iterations can evaluate the quality of a preconditioner. More iterations indicate a poor preconditioner, and fewer iterations indicate a good preconditioner. The $It^{(\iota-1)} \leq \mu$ indicates $B^{(\iota-1)}$ is an effective preconditioner for $A^{(\iota-1)}x^{(\iota-1)} = b^{(\iota-1)}$. In addition, the structure of these matrices is very similar during Newton's iteration, and the preconditioner does not need to approximate the inverse of the matrix exactly. So the preconditioner $B^{(\iota-1)}$ can also be applied to the Jacobian system $A^{(\iota)}x^{(\iota)} = b^{(\iota)}$. The proposed method can improve the parallel performance of the solver by reducing the number of SETUP calls and reducing the proportion of low parallel speedup in the solver.

Finally, we discuss the impact of the threshold μ on performance. If μ is too small, the number of SETUP calls will not be significantly reduced, which will not significantly improve the performance of the solver. In particular, ASMSP degenerates to standard MSP when $\mu = 0$. Conversely, if μ is too large, too few SETUP calls can also affect the performance, due to the dramatic increase in the number of iterations. Usually, a suitable μ is determined by numerical experiments.

4 A multicolor GS based on adjacency matrix

It is well-known that the GS algorithm, compared to the Jacobi algorithm, exploits most updated values in the iterative process. Therefore, the GS algorithm brings a better convergence. However, it is essentially sequential and cannot be easily parallelized. A popular red-black GS (also known as multicolor GS) parallel algorithm has attracted a lot of attention (Saad 2003). Unfortunately, the algorithm is designed based on structured grids and is not compatible with unstructured grids.

A hybrid approach that combines the Jacobi and GS methods can be applied, but its convergence rate also deteriorates with respect to higher parallelism. In order to overcome the limitations of the traditional red-black GS algorithm, a multicolor GS algorithm based on the coefficient matrix of strong connections has been proposed and analyzed in Zhao et al. (2022). This paper proposes a multicolor GS algorithm from the algebraic point of view. The proposed method yields the same convergence behavior as the corresponding single-threaded algorithm; moreover, it obtains good parallel performance when using a lot of threads on GPUs.

4.1 Adjacency graph and algorithm principles

The notion of an adjacency graph needs to be introduced to implement the multicolor GS algorithm algebraically. An adjacency graph corresponds to a sparse matrix, which reflects the nonzero pattern of the matrix, i.e., the nonzero entries of the matrix reflect the connectivity relationship between the vertices in the adjacency graph.

We develop a multicolor GS algorithm that can be applied to symmetric and nonsymmetric matrices. For simplicity, assuming that the sparse matrix $A \in \mathbb{R}^{n\times n}$ is a symmetric matrix. Let $G_A(V, E)$ be the adjacency graph corresponding to the matrix $A = \left(a_{ij}\right)_{n\times n}$. Here $V = \left\{v_1, v_2, \dots, v_n\right\}$ and $E = \left\{\left(v_i, v_j\right) : \forall\, i \neq j, a_{ij} \neq 0\right\}$ are the vertices and edges sets, respectively. It is easy to know that each nonzero entry a_{ij} on the off-diagonal of A corresponds to an edge (v_i, v_j).

Here, we give the principles for designing a multicolor GS algorithm in this paper:

(i) The vertices set V is divided into g subsets: $V = V_1 \cup V_2 \cdots \cup V_g$;
(ii) Any two subsets are disjoint: $V_i \cap V_j = \emptyset$, $i \neq j,\ i,j = 1, 2, \dots, g$;
(iii) Vertices in any subset are not connected by edges: $a_{ij} = a_{ji} = 0,\ \forall\, v_i, v_j \in V_\ell,\ \ell = 1, 2, \dots, g$;
(iv) The number of subsets, g, should be as small as possible.

It is obvious that the difficulty of grouping and parallel granularity increase as the number of groups g decreases. In fact, the red-black GS algorithm satisfies the conditions for $g = 2$. In particular, when $g = n$, it degenerates to the classic GS algorithm.

4.2 multicolor GS algorithm

We define the adjacency matrix S corresponding to the adjacency graph $G_A(V, E)$. Its diagonal entries are zero and off-diagonal entries are

$$S_{ij} = \begin{cases} 1, & \text{if } a_{ij} \neq 0, \\ 0, & \text{if } a_{ij} = 0, \end{cases} \quad \forall\, i,j = 1, 2, \dots, n,\ i \neq j, \tag{23}$$

where $S_{ij} = 1$ denotes an adjacent edge between v_i and v_j, and $S_{ij} = 0$ denotes no adjacent edge between v_i and v_j. To describe the multicolor GS algorithm, some notations are introduced in Table 1.

Below, we first present a (greedy) splitting algorithm for the set of vertices V based on the adjacency matrix S, denoted as VerticesSplitting (see Algorithm 4.1). Then we give the vertices grouping algorithm of matrix A, denoted as VerticesGrouping (see Algorithm 4.2). Finally, a multicolor parallel GS method is presented in Algorithm 4.3, denoted as PGS-MC.

Table 1 The definitions and explanations of some notations

Notation	Definition and explanation
S_i	$S_i = \left\{j : S_{ij} \neq 0, j = 1, 2, \dots, n\right\}$
	S_i denotes the vertex set that is connected to the vertex v_i
$\overline{S}_i$	$\overline{S}_i = \left\{j : j \in S_i \cup \{i\}\right.$ and color of j is undetermined$\left.\right\}$
	$\overline{S}_i$ denotes the vertex set that is connected to the vertex v_i (including v_i) and whose colors are undetermined
$\widehat{S}_i$	$\widehat{S}_i = \left\{j : j \in W_i\right.$ and color of j is undetermined$\left.\right\}$, where $W_i := \left\{j : \forall\, k \in S_i, j \in S_k / (S_i \cup \{i\})\right\}$
	$\widehat{S}_i$ denotes the vertex set that is the next connected to the vertex v_i (the vertices on "the second circle") and whose colors are undetermined
$\lvert S_i\rvert$	$\lvert S_i\rvert = \sum_{j\in S_i} 1$
	The cardinality $\lvert S_i\rvert$ denotes the number of entries in the set S_i

Algorithm 4.1 VerticesSplitting method

```
Require: $V$, $S$;
Ensure: $W$, $\overline{W}$.
1: Let $W = \varnothing$, $\overline{W} = \varnothing$, $\widehat{W} = \varnothing$;
2: while $V \neq \varnothing$ do
3:     if $\widehat{W} \neq \varnothing$ then
4:         Any take $v_i \in \widehat{W}$ and $|S_i| \geq |S_j|$, $\forall v_i, v_j \in \widehat{W}$;
5:     else
6:         Any take $v_i \in V$ and $|S_i| \geq |S_j|$, $\forall v_i, v_j \in V$;
7:     end if
8:     if $v_i$ is not connected to any vertices in $W$ (i.e., $S_{ij} = 0, \forall j \in W$) then
9:         $W = W \cup v_i$, $V = V/v_i$;
10:        if $v_i \in \widehat{W}$ then
11:            $\widehat{W} = \widehat{W}/v_i$;
12:        end if
13:        $\overline{W} = \overline{W} \cup S_i$, $V \leftarrow V/S_i$, $\widehat{W} = \widehat{W} \cup \widehat{S}_i$;
14:    else
15:        $\overline{W} = \overline{W} \cup v_i$, $V \leftarrow V/v_i$;
16:        if $v_i \in \widehat{W}$ then
17:            $\widehat{W} = \widehat{W}/v_i$;
18:        end if
19:    end if
20: end while
```

Algorithm 4.2 VerticesGrouping method

```
Require: $V$, $S$;
Ensure: $\{V_1, V_2, \ldots, V_g\}$.
1: Set $g = 0$;
2: while $V \neq \varnothing$ do
3:     $g = g + 1$;
4:     Get $V_g$ and $\overline{V}_g$ by calling Algorithm 4.1;
5:     Let $V = \overline{V}_g$;
6: end while
```

In our PGS-MC method, it is worth noting that $G_A(V, E)$ can be split into g subgraphs $G_{A_\ell}(V_\ell, E_\ell)$ by calling Algorithm 4.2, and the adjacency matrice corresponding to these subgraphs are S_ℓ ($\ell = 1, \ldots, g$). It is easy to know that the submatrix A_ℓ (corresponding to the subgraph $G_{A_\ell}(V_\ell, E_\ell)$) is a diagonal matrix. In summary, the proposed method starts from the adjacency matrix of the coefficient matrix and designs a vertices grouping algorithm. This method can run in parallel within the same group. Moreover, the proposed method can be applied to the AMG methods, and the numerical experiments in the next section also present its parallel performance.

Algorithm 4.3 PGS-MC method

```
Require: $A, x, b$;
Ensure: $x$.
1: Create the vertices set $V$ and the adjacency matrix $S$ using the matrix $A$ and the formula (23);
2: Generate independent vertices subset $V_\ell$ ($\ell = 1, \ldots, g$) by calling Algorithm 4.2;
3: Use $V_\ell$ to split the matrix $A$ into submatrix $A_\ell$;
4: for $\ell = 1, \ldots, g$ do
5:     Call classical GS algorithm in parallel for submatrix $A_\ell$;
6: end for
```

5 Numerical experiments

In this section, benchmark problems based on SPE1, SPE5, and SPE10 (Odeh 1981; Killough and Kossack 1987; Christie and Blunt 2001) are considered to demonstrate the performance of the proposed methods. Note that the SPE1 and SPE10 problems can be solved using a black oil framework as well; but we solve them using the compositional simulator OpenCAEPoro. It is worth pointing out that, in our simulator, the traditional hydrostatic equilibrium method is used to calculate the initial reservoir conditions; see (Schlumberger 2017) for more details.

In the ASMSP-GMRES method, the Unsmoothed Aggregation AMG (UA-AMG) method is used to approximate the inverse of the pressure coefficient matrix, where the aggregation strategy is the so-called nonsymmetric pairwise matching aggregation (NPAIR) (Napov and Notay 2012), the cycle type is the V-cycle, the smoothing operator is PGS-SCM, the degree of freedom of the coarsest space is set to be 10000, and the coarsest space solver is a direct solver. For

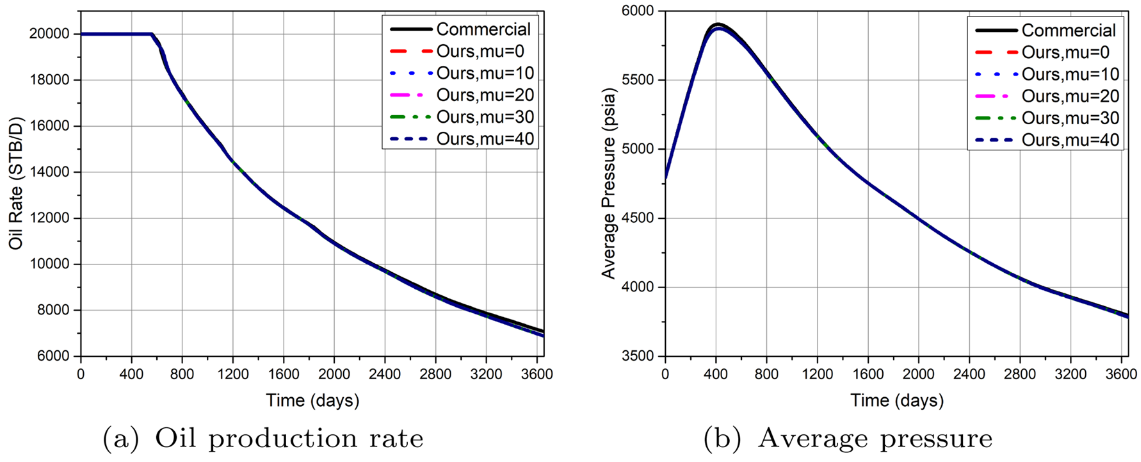


Fig. 2 Field oil production rate and average pressure for the modified SPE1 problem on GPUs

Table 2 SetupCalls, SetupRatio, Iter, SolverTime (s), and Speedup of the different μ for the SPE1 problem

Solvers	μ	SetupCalls	SetupRatio	Iter	SolverTime	Speedup
ASMSP-GMRES-SEQ	0	1178	14.48%	17486	6094.14	–
ASMSP-GMRES-CUDA	0	1177	61.98%	18343	885.07	6.89
	10	1031	61.26%	18529	867.66	7.02
	20	**236**	**50.45%**	**20582**	**749.23**	**8.13**
	30	52	45.46%	22607	750.45	8.12
	40	35	43.47%	24217	778.65	7.83

The results with the best performance are shown in bold

the restarted GMRES(m) method, the restarting number m is 30, the maximum number of iterations *MaxIt* is 1000, and the tolerance for relative residual *tol* is 10^{-5}. Here, Newton method's tolerance is 10^{-3}.

To better evaluate the performance of the proposed methods, we also test the same problems with a commercial simulator (2020 version) for comparison. In the commercial simulator, the default solving method and parameters are used (i.e., Newton method's tolerance is 10^{-3}, and ILU(0)-BiCGstab solver's tolerance is 10^{-3} and the maximum number of iterations is 400). Here, we compare the experimental results of the GPU version for commercial and our simulators. The numerical experiments are tested on a machine with Intel Xeon Platinum 8260 CPU (32 cores, 2.40GHz), 128GB DRAM, and NVIDIA Tesla T4 GPU (16GB Memory).

5.1 The modified SPE1 problem

The three-phase SPE1 example (Odeh 1981) is a benchmark problem for testing ten-year dynamic simulations of immiscible gas flooding (one gas injection well and one oil production well). The initial reservoir state is unsaturated. Given the initial oil-gas and oil-water interface depth, as the reservoir pressure decreases, the gas will gradually dissolve into the oil, and this process will affect the stability of the simulator. The horizontal direction of the oil field

Table 3 NumTSteps, NumNSteps, Iter, AvgIter, TotalTime (s), and Speedup comparisons of the commercial and our simulators for the SPE1 problem

Simulators	μ	NumTSteps	NumNSteps	Iter	AvgIter	TotalTime	Speedup
Commercial	–	410	838	92373	110.2	3382.00	–
Ours	0	267	1177	18343	15.6	2414.36	1.40
	10	267	1177	18529	15.7	2344.43	1.44
	20	**267**	**1179**	**20582**	**17.5**	**2228.08**	**1.52**
	30	267	1178	22607	19.2	2229.42	1.52
	40	267	1178	24217	20.6	2339.99	1.45

The results with the best performance are shown in bold

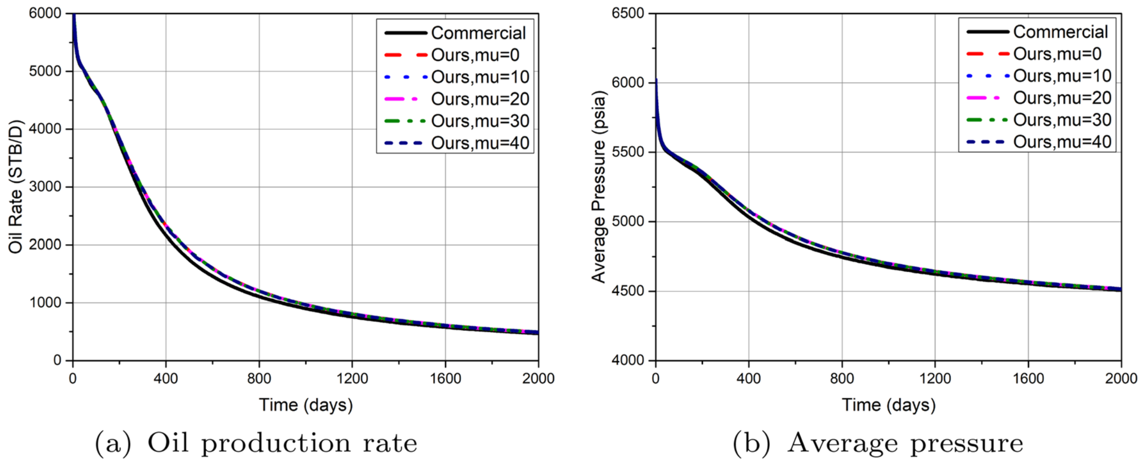


Fig. 3 Field oil production rate and average pressure of the SPE10 problem on GPUs

Table 4 SetupCalls, SetupRatio, Iter, SolverTime (s), and Speedup of the different μ for the two-phase SPE10 problem

Solvers	μ	SetupCalls	SetupRatio	Iter	SolverTime	Speedup
ASMSP-GMRES-SEQ	0	219	12.40%	4252	4795.04	–
ASMSP-GMRES-CUDA	0	219	55.11%	5073	716.08	6.70
	10	172	51.07%	5081	656.37	7.31
	20	118	41.36%	5863	630.31	7.61
	30	**65**	**32.92%**	**6043**	**566.68**	**8.46**
	40	51	27.87%	7055	610.40	7.86

The results with the best performance are shown in bold

is a square with side length of 10000 (ft) and the vertical thickness is 100 (ft). The grid size of the original problem is a $10 \times 10 \times 3$ orthogonal grid, and we refine it to get a grid of $80 \times 80 \times 40$. Here, we perform numerical tests on the refined grid. Below, we simulate 10 years (3655.5 days) using the SPE1 example and test the GPU-based parallel performance of the proposed solver.

5.1.1 ASMSP-GMRES-GPU method

We investigate the effects of μ ($\mu = 0, 10, 20, 30$, and 40) on the parallel performance of ASMSP-GMRES-GPU. We first verify the correctness of ASMSP-GMRES-GPU by comparing our results with commercial simulator. Figure 2 shows the field oil production rate and average pressure graphs of five groups μ.

From Fig. 2, we can observe that the field oil production rate and average pressure obtained by our and commercial simulators are consistent, indicating that the correctness of our proposed methods is guaranteed.

In addition, to evaluate the parallel performance of the proposed methods for ASMSP-GMRES-GPU, Table 2 lists the number of SETUP calls (SetupCalls), the ratio of SETUP in the total solution time (SetupRatio), the total number of linear iterations (Iter), the total solver time (SolverTime), and the parallel speedup (Speedup). ASMSP-GMRES-SEQ is the sequential program for reference.

As can be seen from Table 2, we observe the ASMSP-GMRES-CUDA method. As μ increases, both the number of SETUP calls and the ratio of SETUP in the total solution time decrease, and the parallel speedup first increases and then decreases (since the number of linear iterations increases gradually). Compared with ASMSP-GMRES-SEQ, when $\mu = 0$, the speedup of ASMSP-GMRES-CUDA reaches 6.89. In particular, the solution time of ASMSP-GMRES-CUDA with $\mu = 20$ is reduced from 885.07s to 749.23s compared with $\mu = 0$. Simultaneously, the speedup of ASMSP-GMRES-CUDA reaches 8.13 compared to ASMSP-GMRES-SEQ. Therefore, the proposed methods can obtain acceleration effects for GPU architecture.

Finally, Table 3 presents the number of time steps (NumTSteps), the number of Newton iterations (NumNSteps), the number of linear iterations (Iter), the average number of linear iterations per Newton iteration (AvgIter), the total simulator time (TotalTime), and the parallel speedup (Speedup) for the commercial and our simulators, respectively.

It can be seen from Table 3 that the commercial simulator requires more numbers of time steps and linear iterations, as well as more simulation time, compared to our simulator. They yield the average number of linear iterations per Newton iteration of 110.2, about 7 times as ours (when $\mu = 0$). When $\mu = 0$, the speedup of our simulator achieves 1.40 compared to the commercial simulator. When $\mu = 20$, the minimum simulation time is 2228.08s, and the speedup is 1.52. This indicates that the proposed methods can improve parallel performance. Finally, it is worth noting that we only parallelize the linear solver in our simulator, while the rest of the simulator is still sequential.

5.2 The SPE10 problem

The two-phase SPE10 (Christie and Blunt 2001) benchmark problem with strong heterogeneity is tested to demonstrate the effectiveness of the proposed methods. Its model dimensions are $1200 \times 2200 \times 170$ (ft) and the number of grid cells is $60 \times 220 \times 80$ (the total number of grid cells is 1,122,000 and the number of active cells is 1,094,422). In this example, the numerical simulation is carried out for 2000 days. We analyze the effects of μ ($\mu = 0, 10, 20, 30$, and 40) on the parallel performance of ASMSP-GMRES-GPU.

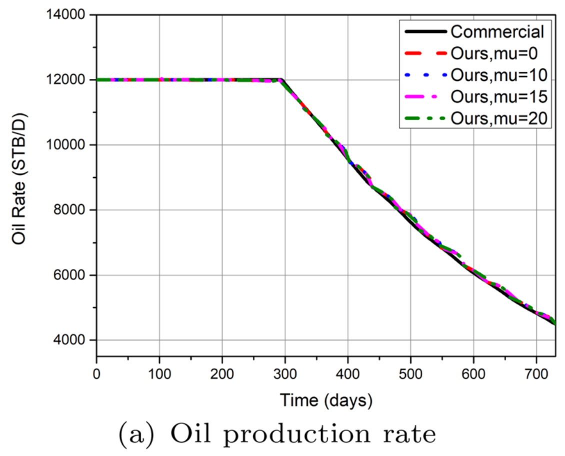
To begin with, we verify the correctness of ASMSP-GMRES-GPU by comparing our results with commercial simulator. The field oil production rate and average pressure graphs of five groups μ are presented in Fig. 3. We note that the field oil production rate and average pressure obtained by our and commercial simulators are consistent, indicating that the proposed methods are corrected.

Furthermore, Table 4 lists the number of SETUP calls (SetupCalls), the ratio of SETUP in the total solution time (SetupRatio), the total number of linear iterations (Iter), the total solver time (SolverTime), and the parallel speedup (Speedup), to assess the parallel performance of the proposed methods. Also, ASMSP-GMRES-SEQ is the sequential program for reference.

Table 5 NumTSteps, NumNSteps, Iter, AvgIter, TotalTime (s), and Speedup comparisons of the commercial and our simulators for the two-phase SPE10 problem

Simulators	μ	NumTSteps	NumNSteps	Iter	AvgIter	TotalTime	Speedup
Commercial	–	1004*	1431*	170276*	119.0*	11034.00*	–
		115	286	157708	551.4	3643.00	–
Ours	0	53	219	5073	23.2	1589.62	2.29
	10	53	219	5081	23.2	1527.12	2.39
	20	53	222	5863	26.4	1534.52	2.37
	30	**53**	**222**	**6043**	**27.2**	**1458.14**	**2.50**
	40	54	221	7055	31.9	1561.90	2.33

The results with the best performance are shown in bold



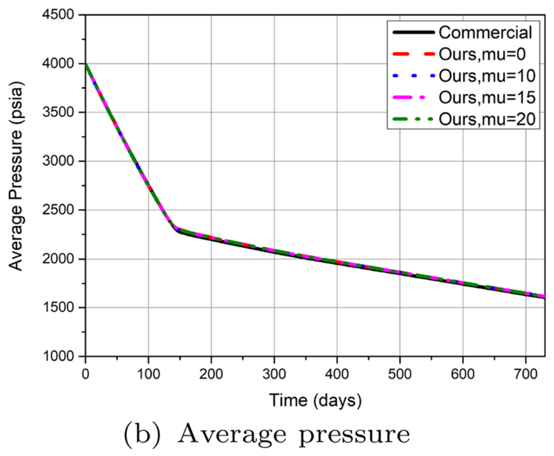


Fig. 4 Field oil production rate and average pressure of the SPE5 problem on GPUs

Table 6 SetupCalls, SetupRatio, Iter, SolverTime (s), and Speedup of the different μ for the SPE5 problem

Solvers	μ	SetupCalls	SetupRatio	Iter	SolverTime	Speedup
ASMSP-GMRES-SEQ	0	389	16.30%	3747	2601.11	–
ASMSP-GMRES-CUDA	0	389	51.43%	3969	341.25	7.62
	10	186	34.59%	4064	324.76	8.01
	15	**44**	**21.72%**	**4508**	**313.23**	**8.30**
	20	12	18.01%	4747	314.50	8.27

The results with the best performance are shown in bold

Table 7 NumTSteps, NumNSteps, Iter, AvgIter, TotalTime (s), and Speedup comparisons of the commercial and our simulators for the SPE5 problem

Simulators	μ	NumTSteps	NumNSteps	Iter	AvgIter	TotalTime	Speedup
Commercial	–	382	748	47027	62.9	2339.00	–
Ours	0	147	389	3969	10.2	2178.78	1.07
	10	147	389	4064	10.4	2159.22	1.08
	15	**147**	**389**	**4508**	**11.6**	**2142.22**	**1.09**
	20	147	389	4747	12.2	2143.47	1.09

The results with the best performance are shown in bold

From Table 4, we observe the ASMSP-GMRES-CUDA method. As μ increases, both the number of SETUP calls and the ratio of SETUP in the total solution time decrease, and the parallel speedup first increases and then decreases. Compared with ASMSP-GMRES-SEQ, when $\mu = 0$, the speedup of ASMSP-GMRES-CUDA reaches 6.70. In particular, the solution time of ASMSP-GMRES-CUDA with $\mu = 30$ is reduced from 716.08s to 566.68s compared with $\mu = 0$. Simultaneously, the speedup of ASMSP-GMRES-CUDA reaches 8.46 compared to ASMSP-GMRES-SEQ.

Finally, Table 5 presents the number of time steps (NumTSteps), the number of Newton iterations (NumNSteps), the number of linear iterations (Iter), the average number of linear iterations per Newton iteration (AvgIter), the total simulator time (TotalTime), and the parallel speedup (Speedup) for the commercial and our simulators, respectively.

In Table 5, the results marked with superscript $*$ of the commercial simulator indicate that the default parameters are used (i.e., Newton method's tolerance is 10^{-3}, and ILU(0)-BiCGstab solver's tolerance is 10^{-3} and the maximum number of iterations is 400). The commercial simulator does not recommend to change these parameters. The SPE10 test is strongly heterogeneous and the default parameters lead to non-convergence of the solver (i.e., iteration reaching the maximum number of iteration) in many Newton steps. This will increase number of Newton iterations and number of time steps. Hence, we test this example using various combinations of tolerance and MaxIt in the commercial software; the best possible choice in our tests are *tol* $= 10^{-5}$ (consistent with ours) and *MaxIt* $= 2000$. Moreover, according to Table 5, the commercial simulator requires more numbers of time steps, Newton iterations, and linear iterations compared to our simulator. The average number of linear iterations per Newton iteration is 551.4 (over 23 times as ours when $\mu = 0$), and the simulation time is 3643.00s. When $\mu = 0$, the speedup of our simulator achieves 2.29 compared to the commercial simulator. When $\mu = 30$, the minimum simulation time is 1458.14s, and the speedup reaches 2.50. These results show that the parallel performance of the proposed methods outperforms commercial simulators.

5.3 The modified SPE5 problem

The SPE5 (Killough and Kossack 1987) example is a compositional reservoir problem, including six components (C_1, C_3, C_6, C_{10}, C_{15}, and C_{20}), injection well (water alternating gas), and production well. Its reservoir domain is $3500 \times 3500 \times 100$ (ft), the original orthogonal grid is $7 \times 7 \times 3$, and the simulation period is 20 years. Here, to evaluate the performance of the proposed methods for compositional reservoir, we refine the original grid to obtain a $70 \times 70 \times 30$ orthogonal grid, and simulate a period of 2 years. We test the effects of μ ($\mu = 0, 10, 15$, and 20) on the parallel performance of ASMSP-GMRES-GPU.

Firstly, we verify the correctness of ASMSP-GMRES-GPU by comparing our results with the commercial simulator. The field oil production rate and average pressure graphs of different μ are presented in Fig. 4. We can see the difference in the field oil production rate and average pressure obtained by our and commercial simulators are consistent, indicating that the proposed methods are corrected.

Furthermore, Table 6 lists the number of SETUP calls (SetupCalls), the ratio of SETUP in the total solution time (SetupRatio), the total number of linear iterations (Iter), the total solver time (SolverTime), and the parallel speedup (Speedup), to assess the parallel performance of the proposed methods. Also, ASMSP-GMRES-SEQ is the sequential program for reference.

From Table 6, we observe the ASMSP-GMRES-CUDA method. As μ increases, both the number of SETUP calls and the ratio of SETUP in the total solution time decrease, and the parallel speedup first increases and then decreases. Compared with ASMSP-GMRES-SEQ, when $\mu = 0$, the speedup of ASMSP-GMRES-CUDA reaches 7.62. In particular, the solution time of ASMSP-GMRES-CUDA with $\mu = 15$ is reduced from 341.25s to 313.23s compared with

Table 8 Iter and SolverTime (s) of the three solvers for the SPE1, SPE10, and SPE5 problems

Solvers	SPE1		SPE10		SPE5	
	Iter	SolverTime	Iter	SolverTime	Iter	SolverTime
MSP-GMRES-PJAC-NO	20880	920.83	7600	813.61	4387	362.04
MSP-GMRES-PGS-NO	19339	888.69	5855	729.38	4099	336.99
MSP-GMRES-PGS-MC	18343	885.07	5073	716.08	3969	333.36

Table 9 NumVerts and NumColors of each level in the AMG method for the SPE1, SPE10, and SPE5 problems

AMG	SPE1		SPE10		SPE5	
Level	NumVerts	NumColors	NumVerts	NumColors	NumVerts	NumColors
0	256002	3	1094426	4	147001	3
1	64002	3	218724	8	36751	6
2	19202	3	62320	9	9800	–
3	6402	–	19660	10	–	–
4	–	–	6646	–	–	–

Table 10 NumVerts and NumColors of each level (excluding the coarsest grid) in the AMG method using five vertices sequences for the SPE10 problem

Level	NumVerts	NumColors				
		Seq1	Seq2	Seq3	Seq4	Seq5
0	1094426	4	4	8	7	7
1	218724	8	7	8	9	8
2	62320	9	5	9	9	9
3	19660	10	5	10	9	9

$\mu = 0$. Simultaneously, the speedup of ASMSP-GMRES-CUDA reaches 8.30 compared to ASMSP-GMRES-SEQ.

Finally, Table 7 presents the number of time steps (NumTSteps), the number of Newton iterations (NumNSteps), the number of linear iterations (Iter), the average number of linear iterations per Newton iteration (AvgIter), the total simulator time (TotalTime), and the speedup (Speedup) for the commercial and our simulators, respectively.

According to Table 7, the commercial simulator requires more numbers of time steps, Newton iterations, and linear iterations compared to our simulator. The average number of linear iterations per Newton iteration is 62.9 (over 6 times as ours when $\mu = 0$), and the simulation time is 2339.00s. When $\mu = 0$, the speedup of our simulator achieves 1.07 compared to the commercial simulator. When $\mu = 15$, the minimum simulation time is 2142.22s, and the speedup reaches 1.09. Finally, we discuss the reasons for the low speedup in this example. As the number of components increases in the compositional model, the complexity of the kernel algorithms increases. Moreover, we only parallelize the linear solver in our simulator, while the rest of the simulator is still sequential. As a result, the parallel solver time is only about 15% of the total simulation time in this example. Hence the linear solver part is not the main computational bottleneck.

5.4 PGS-MC method

To investigate the convergence behavior and parallel performance of the proposed PGS-MC method, we compared it with the parallel GS and Jacobi methods based on natural ordering, denoted as PGS-NO and PJAC-NO, respectively. The MSP-GMRES method is used as a solver for petroleum reservoir simulation. In the MSP preconditioner, the smoothing operator of the AMG method uses PJAC-NO, PGS-NO, and PGS-MC, denoted as MSP-GMRES-PJAC-NO, MSP-GMRES-PGS-NO, and MSP-GMRES-PGS-MC, respectively. Below, we test the GPU-based parallel performance of the three solvers for the SPE1, SPE10, and SPE5 examples.

Table 8 presents the total numbers of linear iterations (Iter) and total solver time (SolverTime) of the three solvers for the SPE1, SPE10, and SPE5 examples. We can observe that MSP-GMRES-PJAC-NO requires more iterations and MSP-GMRES-PGS-MC requires less number of iterations. This shows that MSP-GMRES-PGS-MC produces the same convergence behavior as the corresponding single-thread algorithm [this conclusion is also confirmed by the OpenMP version (Zhao et al. 2022)]. We observed that the solution time of MSP-GMRES-PGS-MC is only slightly less than that of MSP-GMRES-PGS-NO. This is because MSP-GMRES-PGS-MC reduces parallel granularity on coarse levels and increases SETUP time. On the other hand, based on our previous numerical tests, if the number of threads increases, the PGS-NO algorithm usually takes more and more iterations, while PGS-MC is more robust.

In addition, we give the number of colors produced by PGS-MC. Table 9 shows the number of vertices (NumVerts) and the number of colors (NumColors) on each level in the AMG method. It can be seen from Table 9 that the number of colors increases when the number of AMG levels increases. This is due to the coarse levels usually associated with the denser coefficient matrices, which makes multicolor grouping more difficult.

Finally, to check the dependence of the coloring algorithm on the ordering of the vertices, we compare the effects of five numbering sequences on the coloring results. The five numbering sequences are denoted as Seq1, Seq2,..., and Seq5, respectively. Seq1 is the natural order (from small to large, default choice), and Seq2 is the reverse order (from large to small). The last three sequences Seq3–Seq5 are random sequences, generated by a random function. Table 10 lists the number of colors generated by five vertices sequences for the SPE10 problem. According to Table 10, the different vertex numbering produces different colors, which indicates that our coloring algorithm depends on vertex numbering. If too many colors are generated, the parallel granularity will be reduced. In future work, it is worth further studying.

6 Conclusions

In this work, we developed a parallel multistage preconditioner for the system of linear algebraic equations arising from the fully implicit approach for the compositional model. We proposed an efficient multistage preconditioner with an adaptive SETUP to improve the parallel performance of the preconditioner. Furthermore, we developed an improved parallel GS algorithm based on the adjacency matrix. This algorithm can be applied to the smoothing operator of the AMG methods and yields the same convergence behavior as the corresponding single-threaded algorithm. We believe the proposed parallel solver framework will provide a feasible approach to the efficient numerical solution of various application problems. In the future, we will further improve the proposed solver and parallelize our compositional simulator OpenCAEPoro.

Acknowledgements This work was supported by the Postgraduate Scientific Research Innovation Project of Hunan Province (No. CX20210607) and Postgraduate Scientific Research Innovation Project of Xiangtan University (No. XDCX2021B110). Li and Zhang were partially supported by the National Science Foundation of China (No. 11971472). Feng was partially supported by the Excellent Youth Foundation of SINOPEC (No. P20009). Shu was partially supported by the National Science Foundation of China (No. 11971414).

Declarations

Conflict of interest On behalf of all authors, the corresponding author states that there is no conflict of interest.

References

Aavatsmark, I.: An introduction to multipoint flux approximations for quadrilateral grids. Comput. Geosci. **6**, 405–432 (2002). https://doi.org/10.1023/A:1021291114475

Aavatsmark, I., Eigestad, G.T., Mallison, B.T., Nordbotten, J.M.: A compact multipoint flux approximation method with improved robustness. Numer. Methods Partial Differ. Equ. **24**(5), 1329–1360 (2008). https://doi.org/10.1002/num.20320

Al-Shaalan, T.M., Klie, H.M., Dogru, A.H., Wheeler, M.F.: Studies of robust two stage preconditioners for the solution of fully implicit multiphase flow problems. In: SPE Reservoir Simulation Symposium (2009). https://doi.org/10.2118/118722-MS

Aziz, K., Settari, A.: Petroleum reservoir simulation, p. 687. Applied Science Publishers, London (1979)

Brandt, A., Mccormick, S.F., Ruge, J.W.: Algebraic multigrid (AMG) for sparse matrix equations. In: Sparsity and its applications, pp. 257–284. Cambridge University Press, Cambridge (1984)

Cao, H., Tchelepi, H.A., Wallis, J.R., Yardumian, H.E.: Parallel scalable unstructured CPR-Type linear solver for reservoir simulation. In: SPE Annual Technical Conference and Exhibition, vol. SPE-96809 (2005). https://doi.org/10.2118/96809-MS

Chen, Z.X., Huan, G.R., Ma, Y.L.: Computational methods for multiphase flows in porous media. Society for Industrial and Applied Mathematics, New York (2006). https://doi.org/10.1137/1.9780898718942

Chen, Z., Liu, H., Yu, S., Hsieh, B., Shao, L.: GPU-based parallel reservoir simulators. In: Domain decomposition methods in science and engineering XXI, pp. 199–206. Springer, Cham (2014)

Christie, M.A., Blunt, M.J.: Tenth SPE comparative solution project: a comparison of upscaling techniques. SPE Reserv. Eval. Eng. **4**(04), 308–317 (2001). https://doi.org/10.2118/72469-PA

Coats, K.H.: An equation of state compositional model. Soc. Petrol. Eng. J. **20**(05), 363–376 (1980)

Collins, D.A., Nghiem, L.X., Li, Y.K., Grabonstotter, J.E.: An efficient approach to adaptive-implicit compositional simulation with an equation of state. SPE Reserv. Eng. **7**(02), 259–264 (1992). https://doi.org/10.2118/15133-PA

Courant, R., Friedrichs, K., Lewy, H.: Über die partiellen differenzengleichungen der mathematischen physik. Mathematische Annalen **100**(1), 32–74 (1928). https://doi.org/10.1007/BF01448839

Esler, K., Gandham, R., Patacchini, L., Garipov, T., Samardzic, A., Panfili, P., Caresani, F., Pizzolato, A., Cominelli, A.: A Graphics Processing Unit-based, industrial grade compositional reservoir simulator. SPE J. **27**(01), 597–612 (2022). https://doi.org/10.2118/203929-PA

Falgout, R.D.: An introduction to algebraic multigrid computing. Comput. Sci. Eng. **8**(6), 24–33 (2006). https://doi.org/10.1109/MCSE.2006.105

Feng, C.S., Shu, S., Xu, J.C., Zhang, C.S.: A multi-stage preconditioner for the black oil model and its OpenMP implementation. In: Lecture Notes in Computational Science and Engineering **98**, 141–153 (2014)

Fussell, L.T., Fussell, D.D.: An iterative technique for compositional reservoir models. Soc. Petrol. Eng. J. **19**(04), 211–220 (1979)

Hu, X.Z., Xu, J.C., Zhang, C.S.: Application of auxiliary space preconditioning in field-scale reservoir simulation. Sci. China Math. **56**(12), 2737–2751 (2013). https://doi.org/10.1007/s11425-013-4737-3

Kang, Z., Deng, Z., Han, W., Zhang, D.: Parallel reservoir simulation with OpenACC and domain decomposition. Algorithms **11**(12), 213 (2018)

Killough, J., Kossack, C.: Fifth comparative solution project: evaluation of miscible flood simulators. In: SPE Symposium on Reservoir Simulation (1987). https://doi.org/10.2118/16000-MS

LeVeque, R.J.: Finite volume methods for hyperbolic problems. Meccanica **39**, 88–89 (2002)

Li, Z., Wu, S.H., Zhang, C.S., Xu, J.C., Feng, C.S., Hu, X.Z.: Numerical studies of a class of linear solvers for fine-scale petroleum reservoir simulation. Comput. Vis. Sci. **18**(2–3), 93–102 (2017). https://doi.org/10.1007/s00791-016-0273-3

Manea, A.M., Almani, T.: A massively parallel algebraic multiscale solver for reservoir simulation on the GPU architecture. In: SPE Reservoir Simulation Conference (2019)

Meyerink, J.A.: Iterative methods for the solution of linear equations based on incomplete block factorization of the matrix. In: SPE Reservoir Simulation Symposium, vol. SPE-12262 (1983). https://doi.org/10.2118/12262-MS

Middya, U., Manea, A., Alhubail, M., Ferguson, T., Byer, T., Dogru, A.: A massively parallel reservoir simulator on the GPU architecture. In: SPE Reservoir Simulation Conference (2021). https://doi.org/10.2118/203918-MS

Napov, A., Notay, Y.: An algebraic multigrid method with guaranteed convergence rate. SIAM J. Sci. Comput. **34**(2), 1079–1109 (2012). https://doi.org/10.1137/100818509

NVIDIA: CUDA C++ programming guide (2022). https://docs.nvidia.com/cuda/pdf/CUDA_C_Programming_Guide.pdf

Odeh, A.S.: Comparison of solutions to a three-dimensional black-oil reservoir simulation problem. J Petrol Technol **33**(01), 13–25 (1981). https://doi.org/10.2118/9723-PA

OpenACC: OpenACC programming and best practices guide (2022). http://www.openacc.org/

OpenCAEPoro: mlticomponent multiphase porous media flow simulator (2022). https://faspdevteam.github.io/OpenCAEPoro

Peaceman, D.W.: Fundamentals of numerical reservoir simulation. In: Developments in petroleum science, p. 190. Elsevier, Amsterdam (1977)

Peng, D., Robinson, D.B.: A rigorous method for predicting the critical properties of multicomponent systems from an equation of state. AIChE J. **23**(2), 137–144 (1977). https://doi.org/10.1002/aic.690230202

Qiao, C.: General purpose compositional simulation for multiphase reactive flow with a fast linear solver. PhD thesis, The Pennsylvania State University (2015)

Quandalle, P., Savary, D.: An implicit in pressure and saturations approach to fully compositional simulation. In: SPE Symposium on Reservoir Simulation (1989)

Saad, Y.: Iterative methods for sparse linear systems, 2nd edn. Society for Industrial and Applied Mathematics, New York (2003). https://doi.org/10.1137/1.9780898718003

Schlumberger: ECLIPSE Technical Description Version 2017.2 (2017). https://www.software.slb.com/products/eclipse

Stüben, K., Clees, T., Klie, H., Lu, B., Wheeler, M.F.: Algebraic multigrid methods (AMG) for the efficient solution of fully implicit formulations in reservoir simulation. In: SPE Reservoir Simulation Symposium (2007). https://doi.org/10.2118/105832-MS

Sudan, H., Klie, H., Li, R., Saad, Y.: High performance manycore solvers for reservoir simulation. In: European Conference on the

Mathematics of Oil Recovery (2010). https://doi.org/10.3997/2214-4609.20144961

Wallis, J.R.: Incomplete gaussian elimination as a preconditioning for generalized conjugate gradient acceleration. In: SPE Reservoir Simulation Symposium (1983). https://doi.org/10.2118/12265-MS

Wallis, J.R., Kendall, R.P., Little, T.E.: Constrained residual acceleration of conjugate residual methods. In: SPE Reservoir Simulation Symposium (1985). https://doi.org/10.2118/13536-MS

Xu, J.C.: Iterative methods by space decomposition and subspace correction. SIAM Rev. **34**(4), 581–613 (1992)

Xu, J.C.: The auxiliary space method and optimal multigrid preconditioning techniques for unstructured grids. Computing **56**(3), 215–235 (1996)

Yang, B., Liu, H., Chen, Z.X.: Accelerating linear solvers for reservoir simulation on GPU workstations. Society for Computer Simulation International 1, 1–8 (2016). https://doi.org/10.22360/SpringSim.2016.HPC.007

Zhang, C.S.: Linear solvers for petroleum reservoir simulation. J. Numer. Methods Comput. Appl. **43**(1), 1–26 (2022). https://doi.org/10.12288/szjs.s2021-0813

Zhao, L., Feng, C.S., Zhang, C.S., Shu, S.: Parallel multi-stage preconditioners with adaptive setup for the black oil model. Comput. Geosci. **168**, 105230 (2022). https://doi.org/10.1016/j.cageo.2022.105230