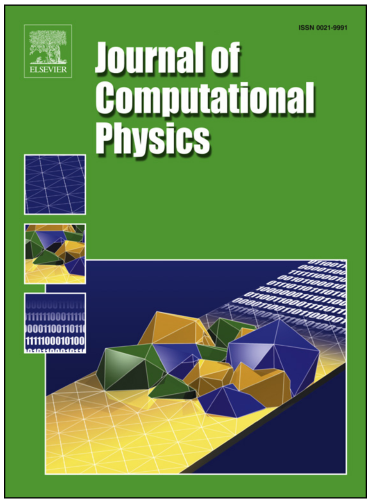


Numerical simulations of polymer flooding process in porous media on distributed-memory parallel computers

He Zhong, Hui Liu, Tao Cui, Zhangxin Chen, Lihua Shen et al.

PII:	S0021-9991(19)30700-4
DOI:	https://doi.org/10.1016/j.jcp.2019.108995
Reference:	YJCPH 108995
To appear in:	*Journal of Computational Physics*
Received date:	12 December 2018
Revised date:	21 September 2019
Accepted date:	26 September 2019

Highlights

- A fully implicit polymer flooding model has been developed under a parallel framework.
- The numerical modeling of polymer retention, inaccessible pore volumes, a permeability reduction and polymer absorption are considered.
- Numerical methods, including discretization schemes, linear solver methods, decoupling algorithm and parallel techniques are introduced.
- Accuracy is verified by comparing with a commercial software.
- Excellent scalability is demonstrated with up to 27 million grid blocks by using up to 2048 CPU cores.

Journal of Computational Physics (2019)

Contents lists available at ScienceDirect

Journal of Computational Physics

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

Numerical Simulations of Polymer Flooding Process in Porous Media on Distributed-memory Parallel Computers

He Zhong[a,*], Hui Liu[a], Tao Cui[b], Zhangxin Chen[a], Lihua Shen[a], Bo Yang[a], Ruijian He[a], Xiaohu Guo[c]

[a]*Chemical and Petroleum Engineering, Schulich School of Engineering, University of Calgary, Calgary T2N 1N4, Canada*
[b]*Academy of Mathematics and Systems Science, Chinese Academy of Sciences*
[c]*Science and Technology Facilities Council, Daresbury Laboratory, Warrington, United Kingdom*

ARTICLE INFO

Article history:
Received September 30, 2019

ABSTRACT

Polymer flooding is a mature enhanced oil recovery technique that has been successfully applied in many field projects. By injecting polymer into a reservoir, the viscosity of water is increased, and the efficiency of water flooding is improved. As a result, more oil can be recovered. This paper presents numerical simulations of a polymer flooding process using parallel computers, where the numerical modeling of polymer retention, inaccessible pore volumes, a permeability reduction and polymer absorption are considered. Darcy's law is employed to model the behavoir of a fluid in porous media, and the upstream finite difference (volume) method is applied to discretize the mass conservation equations. Numerical methods, including discretization schemes, linear solver methods, nonlinearization methods and parallel techniques are introduced. Numerical experiments show that, on one hand, computed results match those from the commercial simulator, Eclipse, Schlumberger, which is widely applied by the petroleum industry, and, on the other hand, our simulator has excellent speedup, which is demonstrated by large-scale applications with up to 27 million grid blocks using up to 2048 CPU cores.

1. Introduction

The emergence of parallel computers compels parallel computation techniques into an array of application areas, including groundwater flow, contamination transport modeling, geothermal engineering, multi-phase flow, carbon dioxide sequestration and nuclear waste storage [1]. Beginning in the mid-1970s, supercomputers were introduced to accelerate reservoir modeling problems through vectorization, and computations could be completed at an advanced speed. However, models and programs had to be reorganized and reworked to take advantage of leveraged computational power through vectorization. Besides, the program performance deteriorated once a CPU number went beyond

*Corresponding author: He Zhong, Tao Cui, Zhangxin Chen
e-mail: hzhong@ucalgary.ca (He Zhong), tcui@lsec.cc.ac.cn (Tao Cui), zhachen@ucalgary.ca (Zhangxin Chen)

a specific number (usually 4, 8 or 16) [2]. Except on a shared memory system, parallel computations can also be carried out on distributed memory clusters. Over the past few decades, significant progress has been made in developing high performance modeling tools for distributed memory systems. However, they have not been widely applied in reservoir simulations.

On the other hand, the demand for modeling capability has increased rapidly in recent years with an increase in computational efforts. More complex geological, physical and chemical features are modeled through reservoir simulations to assess new exploration and production technologies, such as enhanced recovery processes. In addition, the traditional serial simulators have reached their simulation capability limits. The high performance simulation technology has been progressively viewed as an important, alternative modeling approach to solve large-scale simulation problems with multi-million and even multi-billion block models [3]. The reservoir model is divided into several smaller domains that are assigned to different computer processors. The actual time required to finish the simulation is thus reduced because the work load is divided into multiple processors. Furthermore, parallel reservoir simulators are capable to handle larger models than serial simulators.

Currently most oil production comes from mature fields by water flooding. However, water tends to flow through the more permeable formation zones that causes low sweep efficiency and premature water breakthrough. By applying polymer, the viscosity of the water phase is increased, and, as a result, the mobility of the water phase is reduced, which results in a more favorable fractional flow curve and then leads to a more efficient sweeping pattern and reduced viscous fingering. The mobility reduction of the injected water is due to two main effects. First, the viscosity of a polymer solution is higher than that of pure water (the viscosity of a polymer solution increases with raising polymer concentration). Second, the permeability to water is reduced after the passage of a polymer solution through rock materials (the permeability to oil is, however, largely unaffected). Both effects reduce the water mobility while the oil mobility is unaltered.

Polymer flooding holds a bright future because it can improve the area swept efficiency not only in the macro scale but also in the micro scale. The first polymer flooding application was reported in 1964 [4, 5]. The development of polymer flooding boomed in the US during the 1970s and 1980s with several polymer flooding projects [6]. However, it declined in the late 1980s because of low oil prices. During the middle 1990s, polymer flooding was resumed in China to large extent. Especially, oil production from polymer flooding contributed to 22.3% of the total oil production in the Daqing oilfield by 2007 [7, 8]. This significance has attracted the petroleum industry's interest in using reservoir simulators as tools for reservoir evaluation and management to minimize operation costs and increase the process efficiency [9, 10]. Reservoir simulators with special features are needed to represent coupled chemical and physical phenomena present in polymer processes.

Polymer flooding is in reality a miscible process, but it is typically simulated on a field scale using immiscible flow models which use empirical mixture models to account for unresolved miscibility effects. Bondor [11] presented the development of a three-phase, four-component, compressible, finite difference polymer simulator. The model represented a polymer solution as a fourth component that was included in the aqueous phase and was fully miscible with the water phase. Adsorption of polymer was represented as well as the permeability reduction of the water phase. An implicit pressure-explicit saturation (IMPES) procedure was used to solve the coupled system. Lutchmansingh [12] extended this simulator by solving pressure and saturation distributions simultaneously and polymer concentration explicitly. Based on Lutchmansingh's work, Abou-Kassem [13] eliminated non-relevant equations and unknowns by properly ordering the set of all equations and unknowns, thus providing significant savings in CPU time. Chang [14] implemented a third-order finite difference method to capture a physical dispersion effect which is normally smeared by artificial numerical dispersion. An IMPEC (implicit pressure-explicit concentration) scheme was adapted to solve an isothermal, three-dimensional, miscible-flooding compositional model. The simulator is well known as *UTCOMP*. Han [15] improved the solution approach by fully implicit scheme to overcome the performance limitation in symbiosis with the explicit scheme. A field scale chemical flooding model with one million grid blocks were simulated using 128 processors. Based on Han's work, Najafabadi [16] enhanced the phase behavior of surfactant/oil/brine to simulate the realistic salinity gradient effect by assuming no gas phase existing through the formation. Patacchini [17] introduced two-stage flash and two-stage thermodynamics constraints to consider gas/oil/water/microemulsion coexisting four phase problem. Their approach was dependent on a compositional natural-variable formulation. Behzadinasab [18] presented the parallel framework of a comprehensive chemical reservoir simulator that can be effective in running large-scale cases with millions of grid blocks.

To capture fine-scale phenomena and optimize a polymer flooding process, large-scale reservoir simulations with fine-scale grids are required. In this paper, a fully implicit parallel polymer flooding reservoir simulator has been developed to address these issues. The mathematical model of polymer flooding is introduced, including the conservation laws for water, oil and polymer, mechanisms of polymer flooding, and well modelling in the following section that followed by a section where numerical methods are presented. The upstream finite difference (volume) method is applied to discretize the model equations. The standard Newton is applied for their highly nonlinear systems. Linear systems from polymer flooding are ill-conditioned, especially when a reservoir has heterogeneous porosity and permeability. In this case, the linear systems are difficult to solve. In our simulator, a multi-stage preconditioner is employed to speed up the system solution. Parallel implementations are also introduced. Different polymer flooding cases are used to illustrate the accuracy and speedup of our simulator. The results show that this polymer flooding simulator has good speedup and capacity to simulate large-scale reservoir models.

2. Mathematical Model

The two-phase oil and water model is applied, and a temperature change is not considered here. The oil component is assumed to stay in the oil phase, the water stays in the water phase, and polymer only distributes in the water phase. The following sections will present a short summary of all related mathematical models for rock, fluids and well handling.

2.1. Rock Model

When considering porous media at the macro-scale, the flow is governed by volume averaged equations. Each computational block contains both solid and pore space which is filled with fluids, such as gas, oil and water. The percentage of pore space, which is called porosity, is defined as [19]

$$\phi = \frac{V_{pore}}{V_{bulk}}$$

where V_{pore} is the volume of the pore space and V_{bulk} is the volume of a block. Porosity is a function of pressure (and temperature), and it can be modelled by the following equation [10, 19]:

$$\phi(P) = \phi_r + c_r(P - P_r), \tag{1}$$

where c_r is the compressibility factor of the reservoir, P is pressure, and ϕ_r is the reference porosity at the reference pressure P_r.

2.2. Fluid Model

The notion of saturation S_α is introduced to define the ratio of the volume of phase α to the pore space in a block [19]:

$$S_\alpha = \frac{V_\alpha}{V_{pore}}. \tag{2}$$

The saturations of the oil phase (o) and the water phase (w) satisfy the following relationship:

$$S_w + S_o = 1. \tag{3}$$

Darcy's law is applied to handle the relationship among flow rates of a phase, reservoir properties, fluid properties and pressure in a reservoir, which is described as [19]

$$Q = -\frac{\kappa_e A \Delta P}{\mu L}, \tag{4}$$

where A is a cross-sectional area in a flow direction, ΔP is a pressure difference, μ is the viscosity of a fluid, and L is the length of a porous medium in the flow direction. κ_e is the effective permeability for the given phase, which is the product of absolute permeability κ and relative permeability κ_r. κ is defined as a tensor with respect to all the x, y and z directions; mostly, it is a diagonal tensor: $\kappa = (\kappa_x, \kappa_y, \kappa_z)$. Darcy's law can also be rewritten, with Darcy's velocity q,

$$q = \frac{Q}{A} = -\frac{\kappa_e}{\mu}\nabla P. \tag{5}$$

With gravity, the mass of each phase satisfies the following conservation law [19]:

$$\frac{\partial}{\partial t}(\phi S_\alpha \rho_\alpha) = \nabla \cdot \left(\frac{\kappa \kappa_{r\alpha} \rho_\alpha}{\mu_\alpha} (\nabla P_\alpha - \gamma_\alpha \nabla Z) \right) + q_\alpha, \qquad \alpha = w, o \tag{6}$$

where ρ_α is the phase density, q_α is the source term that models the mass changes caused by injection or production wells, γ is the gravity, Z is the depth of a block, and $\kappa_{r\alpha}$ stands for the relative permeability for the α phase. In addition, when polymer exists in the water phase, the mass conservation law for the water phase becomes [19]

$$\frac{\partial}{\partial t}(\phi S_w \rho_w) = \nabla \cdot \left(\frac{\kappa \kappa_{rw} \rho_w}{R_k \mu_{w,e}} (\nabla P_w - \gamma_w \nabla Z) \right) + q_w \tag{7}$$

where R_k is the permeability reduction factor caused by polymer and $\mu_{w,e}$ is the viscosity of a water-polymer solution. The definitions of R_k and $\mu_{w,e}$ will be introduced later.

The water phase pressure, P_w, and the oil phase pressure, P_o, are related by [19]

$$P_c(S_w) = P_o - P_w. \tag{8}$$

The pressure difference is called the capillary pressure, which usually depends on the saturations of the phases in porous media and is measured by lab experiments. If saturation and any phase pressure are known, the other phase pressure can be calculated by the above formula.

2.3. Polymer Model

The flow of polymer is assumed to act as a component dissolved in the water phase, which is modeled by the following equation [19, 20]:

$$\frac{\partial}{\partial t}\Big(\phi S_w \rho_w C_p + (1-\phi) A_d\Big) = \nabla \cdot \left(\frac{\rho_w C_p \kappa \kappa_{rw}}{R_k \mu_{p,e}} (\nabla P_w - \gamma_w \nabla Z) \right) + q_w C_p \tag{9}$$

where C_p is the concentration of the polymer in the water phase and A_d is the polymer adsorbed by the reservoir.

When polymer molecules flow through porous media, part of them are restricted in pores, where only water or brine is allowed to pass by with a reduced mobility. As the polymer solution interacts with the reservoir rock, polymer is adsorbed or desorbed from the rock surface; this mechanism is known as *polymer retention*. There are two mechanisms during the polymer retention process, which are separated as adsorption of the polymer on rock surfaces and entrapment of polymer molecules in small pore space. Both these mechanisms increase the resistance of flow. These effects are modeled by reducing the permeability of the rock to water.

The long chains of polymer molecules can flow into a large pore opening and get trapped when the other end has a smaller opening. Entrapment can also take place when the flow is restricted or stopped. When this happens, the polymer molecules lose their elongated shape and coil up. Desorption of the polymer from the reservoir rock can also take place if sufficient polymer has already been adsorbed above a residual sorption level. It is difficult to quantify what percentage of injected polymer is adsorbed and what percentage is trapped in small pore spaces since only the produced polymer concentration can be measured. Both these mechanisms result in a loss of polymer to the reservoir.

The adsorption process causes a reduction in the permeability of the rock to the passage of the aqueous phase and is directly correlated to the adsorbed polymer concentration. The reduction factor, R_k, is a function of polymer adsorption and the *residual resistance factor* (RRF), which is expressed as [20]

$$R_k = 1.0 + (\mathrm{RRF} - 1.0)\frac{A_d}{A_{d,max}} \tag{10}$$

where A_d is the cumulative adsorption of polymer per unit volume of the reservoir rock and $A_{d,max}$ represents the maximum value of A_d, which denotes the maximum adsorptive capacity of polymer per unit volume of the reservoir rock. Both RRF and $A_{d,max}$ are functions of the reservoir rock permeability.

Assuming equilibrium sorption with the reservoir rock, the sorption phenomenon can be described as a function of polymer concentration C_p only [21]:

$$A_d = f(C_p) \tag{11}$$

This relationship is specified in the form of a table.

Not only is the rock permeability to water reduced after the passage of a polymer solution through porous media, but also the viscosity of the polymer solution is higher than that of pure water. Small concentrations of polymer, on the order of a few hundred to a few thousand ppm (by weight), increase the viscosity of an aqueous solution significantly [22].

The *Todd-Longstaff technique* is used to calculate the effective viscosity that incorporates the effect of physical dispersion at the leading edge of a slug and also the fingering effect at the rear edge of the slug [23]. The viscosity of a fully mixed polymer solution, denoted by $\mu_m(C_p)$, rises as the polymer concentration (C_p) increases. The viscosity of the solution at the maximum polymer concentration is also specified that is denoted by μ_p^0. Then the effective polymer viscosity is taken to be [20]

$$\mu_{p,e} = \left(\mu_m(C_p)\right)^{\omega} \left(\mu_p^0\right)^{1-\omega} \tag{12}$$

where ω is the Todd-Longstaff mixing parameter. The mixing parameter is useful in modeling the degree of segregation between water and the injected polymer solution. If $\omega = 1$, then the polymer solution and water are fully mixied. If $\omega = 0$, the polymer solution is completely segregated from the water. Effect of shear thinning on polymer viscosity is not modeled in the current formulation.

A partially mixed water viscosity is calculated in an analogous manner using the fully mixed polymer viscosity and the pure water viscosity [21, 20]

$$\mu_{w,partial} = \left(\mu_m(C_p)\right)^{\omega} (\mu_w)^{1-\omega} \tag{13}$$

The effective water viscosity is calculated by the partially mixed water viscosity and the effective polymer viscosity as a harmonic average [19]:

$$\frac{1}{\mu_{w,e}} = \frac{\alpha}{\mu_{p,e}} + \frac{1-\alpha}{\mu_{w,partial}} \tag{14}$$

where α is the effective saturation of the injected polymer solution within the total aqueous phase in a block.

The mixing of polymer and water modifies the solution viscosity as well. Since polymer has higher viscosity compared to pure water, no matter which mixing rule is selected, the mixture viscosity increases as a function of polymer concentration in the solution. Two commonly used mixing rules are used, which include a linear mixing rule [19]:

$$\bar{\mu}_w = \beta\mu_p^0 + (1-\beta)\mu_w \tag{15}$$

and a nonlinear mixing rule [21]:

$$\bar{\mu}_w = \left(\mu_p^0\right)^{\beta} (\mu_w)^{1-\beta} \tag{16}$$

where β is a parameter dependent on polymer concentration given by [21]

$$\beta = \frac{C_p}{C_p^0} \tag{17}$$

A higher water viscosity and a reduction in permeability will result in an increase in the resistance to flow, and divert the polymer solution toward areas unswept by water. This mechanism is well-known as *mobility control* [22]. Directly, the water-oil mobility ratio is reduced to close to unity or less. Then the volumetric sweep efficiency is improved and higher oil recovery is achieved compared to conventional water flooding.

As mentioned above, polymer molecules can flow into large pore openings. However, there are also small openings which are not contacted by polymer molecules. To describe this phenomenon, an *inaccessible pore volume* (IPV) is used to measure all the pore space that may not be accessible to polymer molecules [21]. The presence of IPV causes the polymer solution to travel at a greater velocity than inactive tracers embedded in water. This chromatographic effect is modeled by assuming that the IPV is constant for each rock type and either does not exceed the corresponding irreducible water saturation or is independent of the water saturation. The concept of IPV allows a polymer solution to advance and displace oil at a faster rate than predicted on the basis of total porosity.

2.4. Wellbore Models

A numerical simulation of fluid flows in petroleum reservoirs must account for the presence of wells. They supply a set of realistic boundary conditions for computations of pressure distributions [24]. The fundamental task in modeling wells is to model flows into/from a wellbore accurately and to develop accurate well equations that allow the computation of the bottom hole pressure with a given production or injection rate, or the computation of a rate with known pressure [10].

Peaceman [25, 26] associated a steady-state pressure for an actual well with the computed pressure at a grid block through the concept of an equivalent radius r_e. If a well was completed in more than one grid block, a well index (WI) was introduced to account for well pressure losses within the grid blocks due to the radial inflow into the well. The well index depends on the geometry of a grid block, location and orientation of the well segment in that grid block, anisotropic reservoir property and a skin factor [21]:

$$\mathrm{WI} = \frac{2\pi f h f_h \kappa_a}{\ln(r_e/r_w) + s} \tag{18}$$

where f is the well fraction that is evaluated by the angle open to flow and varies due to the well position in a grid block. It equals 1 for a well going approximately through the center of a grid block. h represents a grid block thickness along the well direction, and f_h is the grid block thickness factor. The current completion length in the current grid block is the product of h and f_h. κ_a is the geometric average permeability and estimates the formation's absolute permeability perpendicular to the well direction. s denotes the skin factor, which may also differ from one perforated block to another within the wellbore. This is especially true if different perforation densities and intervals exist within each individual simulation layer. r_w is the wellbore radius. The equivalent radius r_e and formation absolute permeability κ_a are computed according to the wellbore direction and the discretization procedures. For instance, if a well is parallel to the x-direction in a Cartesian grid, then [10, 19]

$$\begin{aligned} r_e &= \frac{2g_f}{\sqrt{\pi}} \frac{(\sqrt{\kappa_z/\kappa_y} h_y^2 + \sqrt{\kappa_y/\kappa_z} h_z^2)^{1/2}}{(\kappa_z/\kappa_y)^{1/4} + (\kappa_y/\kappa_z)^{1/4}} \\ \kappa_a &= \sqrt{\kappa_y \kappa_z} \end{aligned} \tag{19}$$

Similar to the well fraction f, the factor g_f in (19) depends on the geometry of a grid. It equals 0.249 for a well going approximately through the center of a grid block. Detailed information can be found in [21, 27].

The flow rate, $q_{m,\alpha}$, for the α-phase in a perforated grid block m is the product of the well index, the fluid mobility and the drawdown pressure [19, 28]:

$$q_{m,\alpha} = \mathrm{WI}_m \lambda_\alpha \rho_\alpha \left(P_{b,_m} - P_m\right) \tag{20}$$

The wellbore pressure at each grid completion ($P_{b,_m}$) is different from one layer to another, depending on the existing pressure drop in a wellbore. It is calculated by the hydrostatic pressure difference drawn from the average density of the fluid mixture in the wellbore [19]:

$$P_{b,_m} = P_b + \gamma_{well}(z_m - z_b) \tag{21}$$

where γ_{well} is the fluid unit weight which depends on the fluid mixture density in the wellbore and P_b is the reference bottom hole pressure at reference depth z_b [29, 28].

To optimize oil production and to reduce operation costs, various well operations may be employed at any time, such as fixed bottom hole pressure, a fixed oil production rate, a fixed water production rate, a fixed water injection rate, or a fixed liquid production rate [10, 19]. When the fixed bottom hole pressure well operation is applied to a well, the constraint for the well is described as

$$P_b = c, \tag{22}$$

where c is a constant. The fixed water rate condition is the following equation:

$$\sum_m q_{m,w} = q_{c,w}, \tag{23}$$

where $q_{c,w}$ is a constant. For the fixed oil rate production operation, the constraint is

$$\sum_m q_{m,o} = q_{c,o}, \tag{24}$$

where $q_{c,o}$ is a fixed constant. For the fixed liquid production rate operation, the production of oil and water is fixed, and its constraint equation is

$$\sum_m (q_{m,o} + q_{m,w}) = q_{c,l}, \tag{25}$$

where $q_{c,l}$ is a constant.

3. Numerical Methods and Parallelization

In scientific computing research, there are two techniques to handle large-scale models by parallel computing. The first technique is to apply the traditional domain decomposition method, which divides a computational domain to many subregions (subdomains). Each subregion may have overlaps with others and appropriate boundary conditions on the internal boundary that are initiated by the partition process. Then a reservoir simulation model is defined on each subregion. This sub-problem could be much smaller than the original one, but each sub-problem is complete and solved independently. In this case, no parallel linear solver is required and no communication is required during the solution process. Then the solution on the original domain is updated by solutions on the subregions, and communications are required to obtain information from overlapped regions. This method is easy to implement. However, the computation cost raises since iterations are required between different sub-problems to keep the consistency at the internal boundary. The second one is to treat the original computational domain as one domain during the entire simulation period. When parallel computing technique is applied, complicated data structures, algorithms and communications should be designed. Distributed-memory grid, data, matrix, vector, linear solver and preconditioner should be implemented. Also, the order of linear systems could be billions or more if a grid has billions of blocks. The second technique is hard to implement, but it usually has better material balance. In our research, the second approach is employed.

The reservoir model for polymer flooding is highly nonlinear, which is hard to solve analytically. In this paper, numerical solutions are obtained by the fully implicit method thanks to its unconditional stability. Figure 1 shows the details of the solution flowchart, which includes the following steps:

1. Loading model. In this step, a model file is loaded, which contains information for reservoirs, such as permeability, porosity and geometry, for oil, water and polymer, such as density, viscosity, and concentration, for well operations, and for numerical parameters. The model may have hundreds of parameters.
2. Grid generation and distribution. The model file defines a grid, such as dimensions in the x, y, and z directions, sizes of each grid block, and coordinates. Since the simulation is parallel, the grid must be distributed to each processor, and a communication structure must be set up. As section 2.4 described, wellbore contributes as a source term for a grid block when a perforation occurs at the layer. A communication structure is built up based on perforation settings that is critical to guide the message exchanging between processors.
3. Initialization. This step sets the initial pressure, saturations of oil and water, concentration of polymer, bottom hole pressure, porosity and permeability of the reservoir, and other properties, such as relative permeability, density, viscosity, and well index.
4. Time discretization and time step selection. A time step is dynamically selected, which satisfies some conditions:
   4..1 If Newton methods fail, the time step will be cut.
   4..2 In one time step, well operations keep unchanged.
   4..3 The time step has a maximal value, which is set by the model file.
   4..4 The algorithm always attempts to increase a time step to reduce simulation time.
5. Newton iteration. In each time step, an nonlinear system is solved by the Newton method, which converts an nonlinear system to a linear system, $Jx = b$. Inside this step, the properties for the reservoir and fluids must be computed, such as porosity, density and viscosity.
6. Linear iteration. This step solves the linear system $Jx = b$. A proper linear solver, a preconditioner and solution parameters must be chosen, which can be input by the model file. When building Jacobian linear system, the matrix related to a wellbore is placed at the last processor. After a solution is obtained, the well unknowns are broadcast to all other processors, which are essential to calculate drawdown between wellbore and grid blocks, and also source rates.

The MPI (Message Passing Interface) is applied to handle communication among computation nodes. When calculating properties, such as transmissibility, neighboring information is always required. To develop a scalable parallel application, communications should be minimized. In reservoir simulations, the communication pattern is determined by a grid distribution.

Grid partitioning algorithms aim to minimize the idle time and communication flow between different processors by dividing the blocks equally among partitions and minimizing the number of partition spanning edges. The quality of the partitioning plays a crucial role in the performance of applications. Reservoir simulation applies grid-based

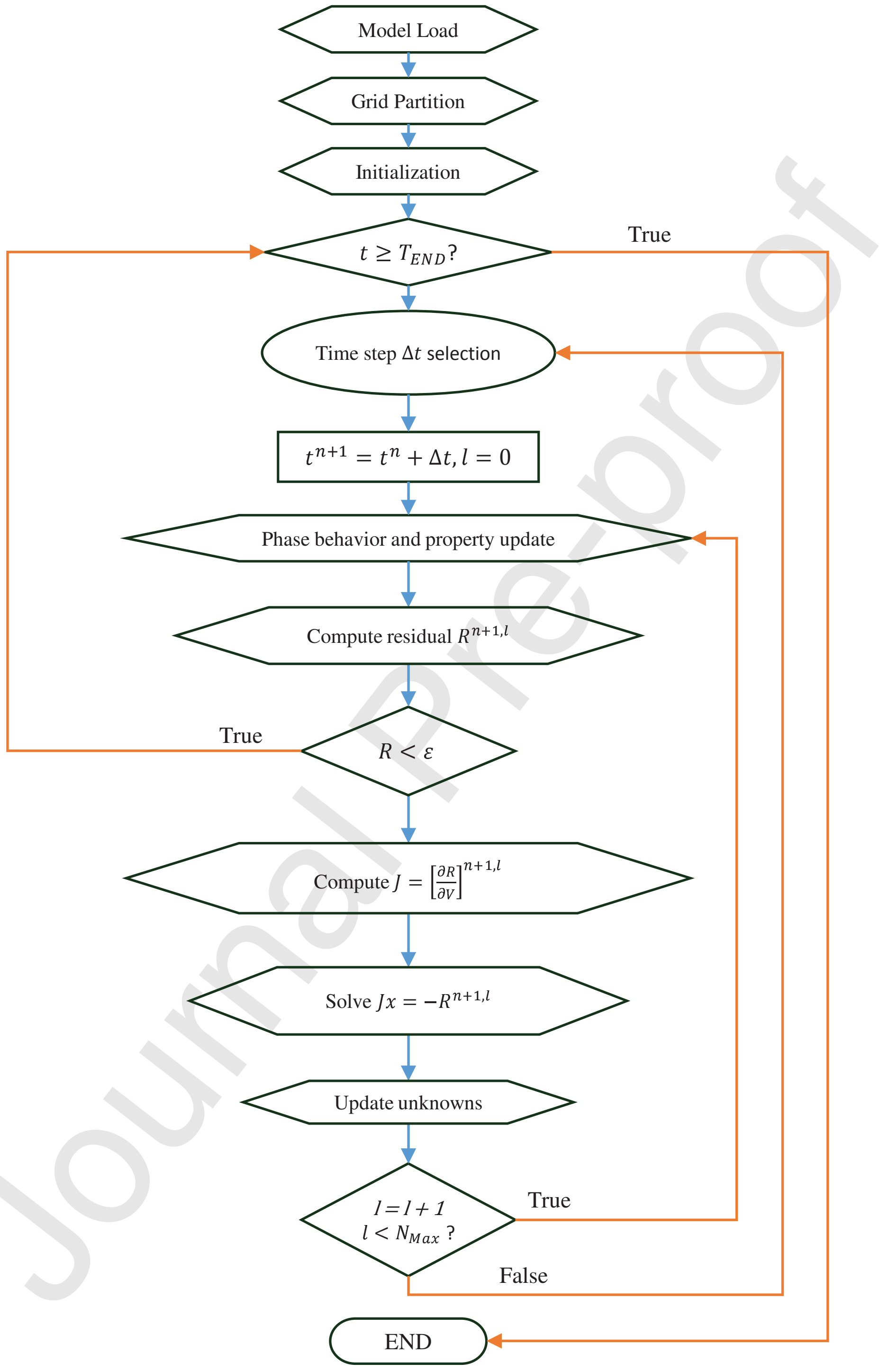

Model Load
Grid Partition
Initialization
$t \geq T_{END}$?
True
Time step Δt selection
$t^{n+1} = t^n + \Delta t, l = 0$
Phase behavior and property update
Compute residual $R^{n+1,l}$
True
$R < \varepsilon$
Compute $J = \left[\frac{\partial R}{\partial V}\right]^{n+1,l}$
Solve $Jx = -R^{n+1,l}$
Update unknowns
$l = l + 1$
$l < N_{Max}$?
True
False
END

Fig. 1. Parallelism protocol of general domain decomposition strategy.

numerical methods, such as the finite volume and finite difference methods, to approximate a governing system. Grid blocks have a higher possibility to communicate with each other when they are closer to each other. Based on this assumption, a modified Hilbert space-filling curve partitioning method was introduced [30] in our in-house simulator, which has shown promising loading balance and excellent speedup. Since the reservoir model is normally upscaled from a geological model, there may exist some grid cells whose porosity and/or permeability are effectively zero. These grid cells are defined as inactive blocks who are excluded from flow simulation. A two-stage method is implemented to handle the work balance problem introduced by inactive blocks. In the first stage, the Hilbert space-filling curve partitioning method is applied on all active blocks. Then inactive blocks is partitioned in a following stage. By two stages, all active and inactive blocks are distributed evenly.

After the grid partitioning subroutine occurs, the subregion topology graph and block level connection lists are set up. The block level connection lists record the geometry information both in the global and local indices. Here, the global index is normally defined by the natural ordering. It labels the grid blocks in the x, y and z directions, respectively. Once the subregion configuration is fixed, the dataset is read and distributed into each processor. A parallel IO algorithm through MPI-IO is implemented to read the data files and to write data files.

A set of data structures have been designed to store distributed data, such as DOF (Degrees of Freedom), VEC, MAT, SOLVER, and SOLVER_PC [31]. A DOF is defined on a grid, which can be defined to store block-based data, such as porosity, density and viscosity, and on a well, which can be defined for bottom hole pressure, a well rate, and a well index. The VEC and MAT are distributed vectors and matrices. Vector operations, such as dot product, and sparse matrix-vector multiplication operations are implemented, which are used to develop parallel linear solvers and preconditioners. The reader refers to our previous paper [31] for more details of parallelization.

3.1. Numerical Discretization

The backward Euler method is applied to discretize a time derivative numerically. The system is then solved at each time step implicitly. The oil phase pressure, water saturation, polymer concentration and well bottom hole pressure are chosen as the primary unknowns, and other unknowns are functions of these primary unknowns.

$$\nabla \cdot \left(\frac{\kappa\kappa_{r\alpha}\rho_\alpha}{R\mu_\alpha} \left(\nabla P_\alpha - \gamma_\alpha \nabla Z\right) \right) = \nabla \cdot \left(\frac{\kappa\kappa_{r\alpha}\rho_\alpha}{R\mu_\alpha} \nabla \Phi_\alpha \right) \tag{26}$$

When fluids move in a reservoir, there may be fluid exchange in two neighboring blocks. The term, transmissibility, is defined to describe the amount of fluid exchange. Here, let d ($d = x, y, z$) be any space direction and A be the area of a face in the d direction; then the transmissibility term $T_{\alpha,d}$ for phase α is defined as

$$T_{\alpha,d} = \frac{\kappa\kappa_{r\alpha}}{R\mu_\alpha}\rho_\alpha \frac{A}{\Delta d}, \tag{27}$$

where Δd is the grid block length along the d direction, κ is the permeability, $\kappa_{r\alpha}$ is the relative permeability, μ_α is the viscosity, and R is 1 for the oil phase and R_k for water and polymer.

The transmissibility is defined on each face of a block. For any two neighboring blocks, since they share a face, the value of the transmissibility term is the same for these two blocks. Different weighting schemes must be applied to average different properties at an interface for them to make a physical sense. The equation (27) can be written into two parts,

$$T_{\alpha,d} = \frac{\kappa A}{\Delta d} \times \frac{\kappa_{r\alpha}}{R\mu_\alpha}\rho_\alpha, \tag{28}$$

The left part includes geometric properties, such as a grid block length Δd and a cross area A, and rock permeability κ. Since it is approximated at an interface, harmonic averaging method is applied. The right part relies on fluid properties, such as μ_α and ρ_α, which is approximated by weighted upstream technique. It means that the value of the right part is from the block that has a higher flow potential. For example, the relative permeability $\kappa_{r\alpha}$ at the interface $(i - 1/2, j, k)$ is defined as [19]

$$(\kappa_{r\alpha})_{i-\frac{1}{2},j,k} = \begin{cases} (\kappa_{r\alpha})_{i,j,k} & \text{if } \Phi_{i,j,k} \ge \Phi_{i-1,j,k} \\ (\kappa_{r\alpha})_{i-1,j,k} & \text{if } \Phi_{i,j,k} < \Phi_{i-1,j,k} \end{cases}. \tag{29}$$

Other higher-order upstream weighting techniques can also be used for relative permeabilities.

When assembling the Jacobian matrix, the partial derivatives of a function with respect to oil phase pressure, water saturation, polymer concentration and bottom hole pressure need to be calculated. In Section 2, analytical equations have been introduced to calculate porosity, viscosity and sink/source term. When calculating the partial derivatives of these properties, analytical expressions can be applied right away with chain rule. However, relative permeabilities and capillary pressure are approximated by user-input table, proper interpolation techniques are required, such as linear interpolation, cubic interpolation or monotone cubic interpolation. In parallel computing environment, communications are required to obtain remote information, and proper index should also be set in the Jacobian matrix.

3.2. Linear Solver

A Jacobian matrix is nonsymmetric and highly ill-conditioned. The Krylov subspace solvers are applied to solve the linear system $Jx = b$. In real application, a preconditioner M is always applied to solve an equivalent linear system $M^{-1}Jx = M^{-1}b$. A family of scalable constrained pressure residual (CPR) methods [32] have been developed to handle linear systems from reservoir simulations.

The system has four unknowns, oil phase pressure (P_o), water saturation (S_w), polymer concentration (C_p) and well bottom hole pressure (P_b), which are vectors. There are two common strategies to arrange the unknown x, which are point-wise and block-wise, respectively. For the point-wise strategy, x is written as

$$x = \begin{pmatrix} P_o \\ S_w \\ C_p \\ P_b \end{pmatrix}, \tag{30}$$

while, for the block-wise strategy, x is written as

$$x = \begin{pmatrix} P_{o,1} \\ S_{w,1} \\ C_{p,1} \\ \cdots \\ P_{o,n} \\ S_{w,n} \\ C_{p,n} \\ P_{b,1} \\ \cdots \\ P_{b,n_\varpi} \end{pmatrix}, \tag{31}$$

where n is the number of grid blocks and n_ϖ is the number of wells. In real simulations, the preconditioner using the block-wise strategy has better convergence than that using the point-wise strategy.

The block-wise strategy for unknown x is adopted in our research. Each grid block has three equations, which are mass conservation laws for oil, water and polymer, respectively, and each well has one equation, which depends on the operation conditions. In this paper, the mass conservation equations are numbered grid block by grid block, which means, in the i-th block, the three equations are numbered as $3 \times (i-1) + 1$, $3 \times (i-1) + 2$, $3 \times (i-1) + 3$. In this case, the Jacobian matrix has the follow structure,

$$J = \begin{pmatrix} J_{11} & \cdots & \cdots & J_{1n} & J_{1\sigma} \\ J_{21} & J_{22} & \cdots & J_{2n} & J_{2\sigma} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ J_{n1} & J_{n2} & \cdots & J_{nn} & J_{n\sigma} \\ J_{\sigma 1} & \cdots & \cdots & J_{\sigma n} & J_{\sigma\sigma} \end{pmatrix}, \tag{32}$$

where $J_{ij} (1 \le i \le n, 1 \le j \le n)$ is a $R^{3\times 3}$ matrix, $J_{\sigma i} (1 \le i \le n)$ is a $R^{n_\varpi \times 3}$ matrix, $J_{i\sigma} (1 \le i \le n)$ is a $R^{3\times n_\varpi}$ matrix, and $J_{\sigma\sigma}$ is a $R^{n_\varpi \times n_\varpi}$ matrix.

3.3. Decoupling Methods

It is well-known that a good decoupling method is critical to the efficient solution of linear systems from reservoir simulations. In general, the decoupling process can be written as,

$$(D^{-1}J)x = D^{-1}b, \tag{33}$$

in which the decoupling method should be effective, easy to obtain and computationally efficient. Many decoupling methods have been developed, such as Quasi-IMPES, True-IMPES [33], Alternate Block Factorization (ABF) [34], and full row sum (FRS)[35] methods. The basic idea of ABF method is to partially decouple the system by reducing the effect of the off-diagonal blocks. It is defined as,

$$D_{abf} = diag(J_{11}, J_{22}, \cdots, J_{nn}, I_{\sigma\sigma}). \tag{34}$$

The block diagonal part of the new Jacobian matrix is identity matrix after applying ABF method.

If the FRS decoupling method is adopted to our model, then it is described,

$$D_{frs}^{-1} = diag(D_1, D_2, \cdots, D_n, I_{\sigma\sigma}), \tag{35}$$

where,

$$D_i = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}. \tag{36}$$

When the FRS method is applied to the Jacobian matrix, it means to add the second and the third rows to the first row.

The Guass-Jordan elimination (Gauss elimination) method is named after Gauss, which has been widely used to solve linear systems. In this paper, it is adopted as a decoupling method. The idea is to convert $[D|J|b]$ to an equivalent linear system $\left[I|\tilde{J}|\tilde{b}\right]$ by Gauss-Jordan elimination method, which is written as following,

$$[D|J|b] \longrightarrow \left[I|\tilde{J}|\tilde{b}\right] \Longrightarrow \tilde{J}x = \tilde{b}, \tag{37}$$

where I denotes an identity matrix. From now on, the notation GJE will be used to represent Gauss-Jordan decoupling method. Here are some notes for GJE decoupling method:

1. The column pivoting technique is applied. During the decoupling process, the rows of the Jacobian matrix are reordered, which, however, does not change the solution.
2. No communication is required, which is friendly to parallel computing.
3. The Jacobian matrix can be adjusted grid block by grid block independently.
4. The row pivoting is not applied. Since if the columns of the Jacobian matrix are reordered, a global reordering will be required, which is expensive and requires lots of MPI communications.
5. The Guass-Jordan decoupling method is more efficient than the ABF method, which requires to calculate the inverse of the diagonal part and the matrix-matrix multiplications.

When the CPR-like preconditioners are applied to reservoir simulations, it is important to keep the pressure matrix positive definite. FRS method helps to enhance this property, from which the CPR-like preconditioners can benefit. A two-stage decoupling methods are also introduced. During the first stage, FRS and ABF methods are applied in sequence, which is noted as FRS-ABF decoupling method. And FRS and GJE methods are used as following stage, which is noted as FRS-GJE method.

3.4. Preconditioners

For the sake of completeness, the CPR-like preconditioners developed in our previous paper are introduced here. If the point-wise strategy for x is applied, the Jacobian matrix J has a clear block structure as

$$J = \begin{pmatrix} J_{PP} & J_{PS} & J_{PC} & J_{P\varpi} \\ J_{SP} & J_{SS} & J_{SC} & J_{S\varpi} \\ J_{CP} & J_{CS} & J_{CC} & J_{C\varpi} \\ J_{\varpi P} & J_{\varpi S} & J_{\varpi C} & J_{\varpi\varpi} \end{pmatrix}, \tag{38}$$

where $J_{PP} \in R^{n\times n}$ is the matrix corresponding to the oil phase pressure, $J_{SS} \in R^{n\times n}$ is the matrix corresponding to the water saturation, $J_{CC} \in R^{n\times n}$ is the matrix corresponding to the polymer concentration, $J_{\varpi\varpi} \in R^{n_\varpi \times n_\varpi}$ is the matrix corresponding to the well bottom hole pressure, and other matrices are coupled terms.

If a restriction operator from x to P_o is defined, its formal formula is written as

$$\Pi_r x = P_o. \tag{39}$$

A prolongation operator Π_p from P_o to x can be defined as

$$\Pi_p P_o = \begin{pmatrix} P_{o,1} \\ \cdots \\ P_{o,n} \\ \vec{0} \\ \cdots \\ \vec{0} \\ \vec{0} \end{pmatrix}. \tag{40}$$

The preconditioning linear system $My = f$ must be solved in each iteration. The CPR-FPF method [32], a three-stage preconditioner, can be described by Algorithm 1, where the first step is to solve an approximate solution using restricted additive Schwarz (RAS) method, the third step is to solve the subproblem by algebraic multi-grid method (AMG), the fifth step is to get an approximate solution again using restricted additive Schwarz method, and the second step and the forth step are to calculate residual. It is well-known that RAS method and AMG method are scalable for parallel computing, so the CPR-FPF method is also scalable. Here we should mention that the setup phase of the AMG method is computationally intense. The sub-problem for each CPU from RAS method is solved by ILUT by default. CPR-FP and CPR-PF preconditioners were also proposed [32].

Algorithm 1 The CPR-FPF Method

```
1: $y = RAS(J)^{-1} f$
2: $r = f - Jy$
3: $y = y + \Pi_p AMG(J_{PP})^{-1} \Pi_r r$
4: $r = f - Jy$
5: $y = RAS(J)^{-1} f$
```

4. Numerical Experiments

4.1. Model Validation

In this section, several examples are presented to validate the application of our simulator (BOS) in different polymer flooding projects. First, a homogeneous, one-dimensional polymer flooding case with 15 blocks is presented. The polymer concentration varies in different production stages. Second, a two-dimensional polymer flood with multiple constraints on the injection wells is considered. Finally, a three-dimensional case with a stratified reservoir is presented. It has a number of layers with significant or complete lateral continuity. The layer permeabilities vary greatly and adjacent layers communicate vertically. Simulation results show that water advances rapidly in highly permeable layers and slowly in tight layers as the driving fluid to displace oil.

Eclipse, Schlumberger, is a fully-implicit, three-phase, three-dimensional, general purpose black oil simulator that can model different chemical EOR (enhanced oil recovery) processes, including polymer and surfactant flooding. It can be run in fully implicit and adaptive implicit modes, and is widely considered as a reference and benchmark standard, which has been successful in reproducing laboratory measurements and large-scale applications. It is widely used in the oil industry to evaluate different production processes. Our simulator will be compared against it to show the accuracy.

4.1.1. 1D Polymer Flooding Model

Here a special case of polymer flood in one-dimension geometry is introduced. The reservoir contains 15 grid blocks along the x-direction where the properties are uniformly distributed at each grid block with porosity $\phi = 0.5$ and intrinsic permeability $\kappa = 100\,mD$. A polymer adsorption curve and water viscosity multiplier are displayed in Figure 2, and the fluid properties are summarized in Table 1.

Water is injected with a fixed constant volume rate at $1000\,m^3/day$ by the injection well located at the first grid block. The production well is perforated through the 15th grid block, where a liquid volume well constraint was

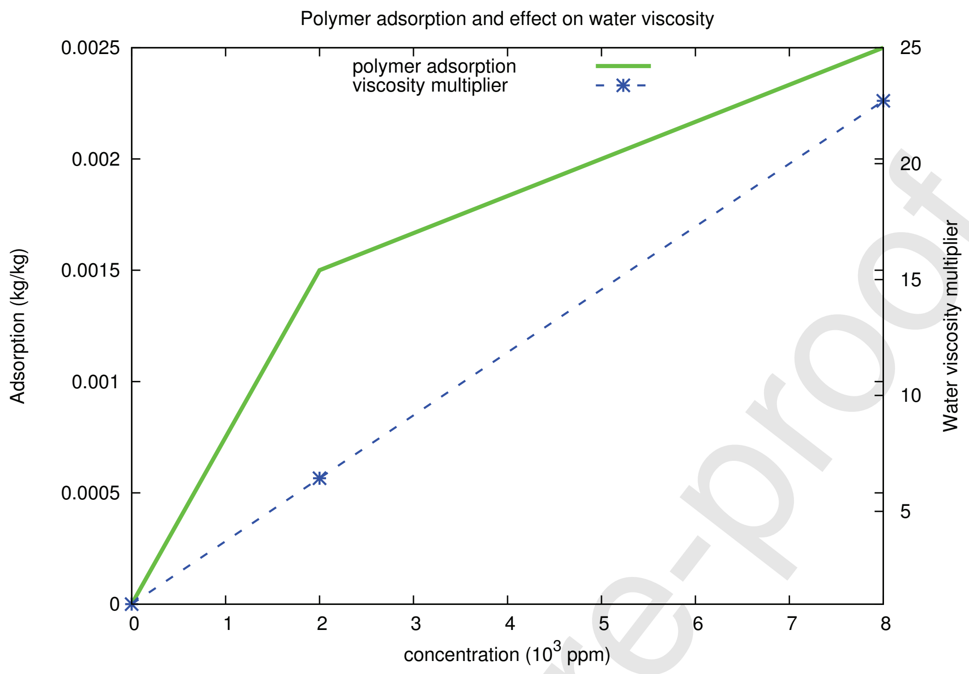


Fig. 2. Polymer adsorption curve and water viscosity multiplier.

Table 1. Simulation input data of 1D case.

Model parameter	Value
Spatial grid size (m)	100
Initial resident water saturation	0.4
Initial polymer concentration (kg/m^3)	0.1
Water	
Mass density (kg/m^3)	1025.18
Formation volume factor (sm^3/rm^3)	1.0
Compressibility (1/Bar)	3.03e-6
Viscosity (cp)	0.5
Oil	
Mass density (kg/m^3)	832.96
Formation volume factor (sm^3/rm^3)	1.0
Compressibility (1/Bar)	1.0e-5
Viscosity (cp)	0.5
Rock	
IPV	0.15
RRF	2.67
Maximum polymer adsorption (kg/kg)	0.0035

operated at 1000 m^3/day. The polymer injection concentration varies along the production process. At the beginning, only pure water is injected for 300 $days$, and then polymer is added with concentration at 6000 ppm to improve the volumetric sweep efficiency. Finally, polymer injection is stopped after 500 $days$ injection, and simulation is terminated after total 1800 $days$.

In Figure 3, the oil and water production rates are compared with those from Schulumberger-Eclipse that are represented by the green line, while our numerical solutions are plotted in red color and line markers. After about 500 *days* of water injection, the water front has broken through, and the oil production rate reduces until the whole reservoir is flooded. A good agreement is found in Figure 3 with some minor difference due to numerical diffusion and different ways to approximate the water phase viscosity.

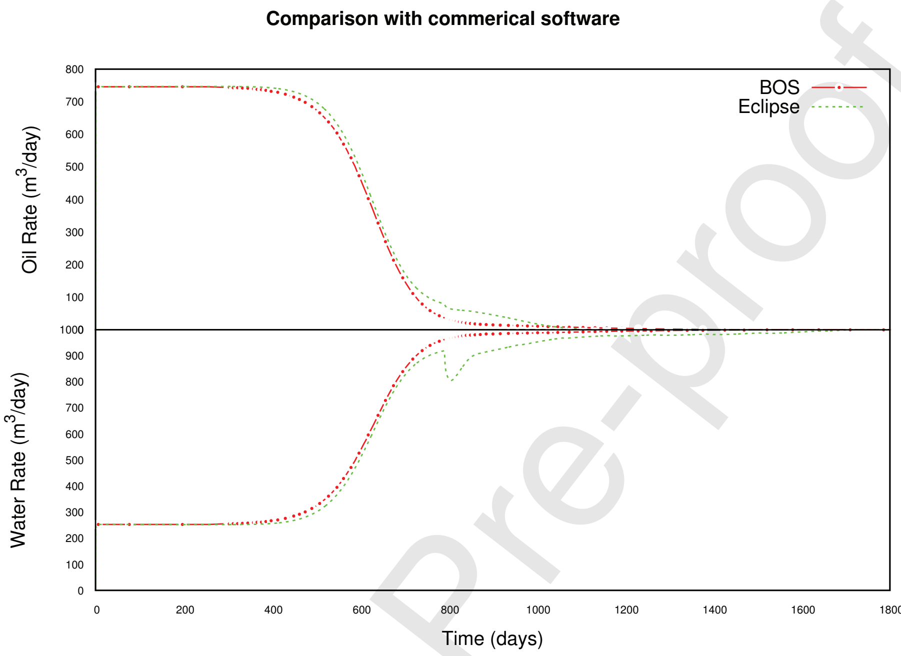
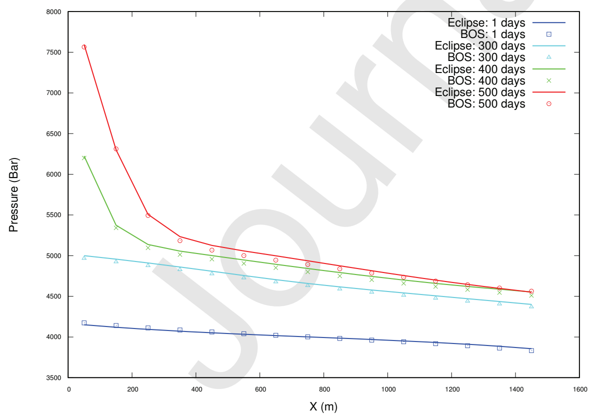


Fig. 3. Comparison of oil and water production rate between BOS and Eclipse on 1D example.



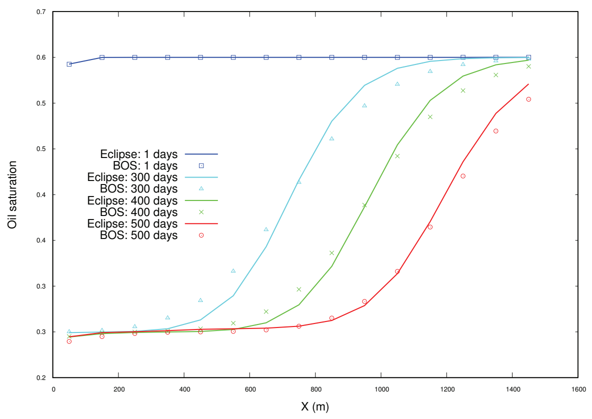


Fig. 4. Pressure profiles at selected time steps.

Fig. 5. Oil saturation profiles at selected time steps.

The pressure and oil saturation profiles after 1, 300, 400 and 500 days are compared with Eclipse at Figure 4 and Figure 5, respectively. The IBM BladeCenter HS23 server is used to carry out the performance comparison tests between our simulator and Eclipse. The server is equipped with Intel Xeon processor E5-2680 and 16 DDR3 VLP memory DIMM slots that support up to 256 GB of DDR memory. The standard Newton iterations converge when the relative residual of the balance equations is less than 0.01. Biconjugate Gradient Stabilized (BICGSTAB) and RAS preconditioner are used to solve the linear system. The initial time step is 20 days. The numerical summaries are shown in Table 2, where the amount of time steps, Newton iterations and linear solver iterations are listed. The number of Newton iterations is a measure of how easy it is for the model to converge. As a rough guide to the efficiency of convergence of a model, which may vary form time step to time step. The average Newton iteration through the simulation are listed by 'Avg N.' column in the table. It takes average around 3.5 Newton iterations to converge for each time step. The other indicator 'Avg L.' shows the efficiency of the Krylov iteration solver and preconditioner. Here around 4.5 iterations are required to solve the linear systems. The other columns, 'Steps', '# Newton', '# Solver' and 'Time (s)', respectively denote the total time step, total Newton iterations, total linear iterations and total CPU time that each case costs. For this one-dimensional case, our simulator takes 0.023 seconds while Eclipse requires 0.51 seconds to finish the simulation if only one processor is used.

Table 2. Numerical summaries for the one dimensional case.

	Steps	# Newton	Avg N.	# Solver	Avg L.	Time (s)
BOS	17	59	3.47	264	4.47	0.023
Eclipse	26	80	3.08	80	1.	0.51

4.1.2. 2D Polymer Flooding Model

Next, a polymer flood case on a 10×10 grid is simulated and the formation is initially fully saturated with oil with porosity $\phi = 0.2$ and intrinsic permeability $\kappa = 50\,mD$. The fluid properties are summarized in Table 3.

Table 3. Simulation input data on 10×10 grid.

Model parameter	Value
Spatial grid size (ft)	75
Initial polymer concentration (lbm/bbl)	0.
Water	
Mass density (lbm/cuft)	64
Formation volume factor (STB/bbl)	1.0
Compressibility (1/psia)	3.03e-6
Viscosity (cp)	0.5
Oil	
Mass density (lbm/ft^3)	52
Formation volume factor (STB/bbl)	1.0
Compressibility (1/psia)	1.0e-5
Viscosity (cp)	2

There is no polymer in the reservoir initially. An injection well locates at the left corner of the grid and a production well locates at the other end of the diagonal. Water and a polymer solution are injected at a maximum injection rate of $200\,STB/day$ with polymer concentration at $50\,lbm/STB$. At the same time, the injection well is constrained with its bottom hole pressure, with no more than $2 \times 10^5\ psia$. There is no constraint on the production rate, but the bottom hole pressure of the production well is fixed at $3999\,psia$. After $200\,days$ production, there is no polymer injected into the formation. The simulation is terminated after total $1700\,days$.

The same case was carried out by Eclipse, and the oil and water production rates are compared in Figure 6 by the red and green colors that represent our simulator and Eclipse, respectively. As Figure 6 shown, the same production curves are found with different simulators.

In order to compare pressure and oil saturation field, three grid blocks are selected as $(6, 8, 1)$, $(5, 5, 1)$ and $(8, 3, 1)$. The pressure and oil saturation at these grid blocks are compared with Eclipse at different time in Figure 7 and Figure 8 respectively. The performance comparison is carried by IBM BladeCenter HS23 server. Only one processor is applied to finish the simulation. Same as the previous case, standard Newton iterations convergence tolerance is selected at 1e-2 and the solver tolerance is 1e-3. The initial time step is 50 days. The CPR preconditioner and BICGSTAB iteration method are combined to solve the linear system. For all these three case, the amount of time steps, Newton

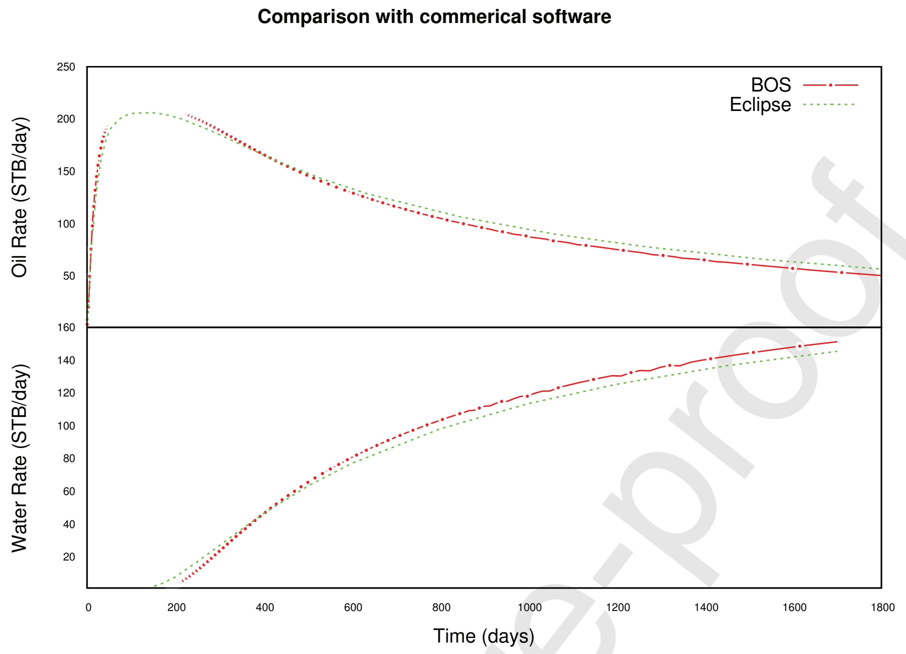


Fig. 6. Comparison of production rate between BOS and Eclipse on a two-dimensional polymer flood.

iterations, linear solver iterations and running time are listed in Table 4. Our simulator and Eclipse cost 0.1773 and 0.42 seconds, respectively.

Table 4. Numerical summaries for the two dimensional case.

	Steps	# Newton	Avg N.	# Solver	Avg L.	Time (s)
BOS	43	76	1.77	230	3.03	0.1773
Eclipse	26	87	3.35	304	3.49	0.42

4.1.3. 3D Polymer Flooding Model

In this case, the two-dimensional case is extended into a three-dimensional problem. The same as in the two-dimensional problem, two vertical wells locate at the two ends of the diagonal of a *xy* plane and act as an injection well and a production well. Both wells have a multiple layer perforation that is different from the two-dimensional case. For multiple perforated layer wells, a reference bottom hole pressure P_b is selected at a datum depth. The fluid density is calculated at that depth and the initial pressure in the reservoir is determined by marching down the reservoir in small steps by recalculating the density at each step. It is treated explicitly through the Newton iteration. This algorithm is more complicated when more than one phase is present. Starting at the datum depth, the hydrostatic pressure for the datum phase can be calculated by marching up and down the reservoir. Pressures in the other phases can then be determined at the contact depths, and then the hydrostatic pressures can be determined throughout the reservoir by marching up and down again. Once the phase pressures are known, the phase saturations can be determined at each depth so that the hydrostatic pressure variation is balanced by the capillary pressure between the phases.

The agreement of our simulator with commercial software is shown in Figure 9 for the oil and water production rates. Besides, the pressure and oil saturation profile are also demonstrated respectively in Figure 10 and Figure 11 for the top layer of the formation. The BICGSTAB and CPR preconditioner are applied. The Newton and linear tolerances are 1e-2 and 1e-3, and the initial time step is 100 days. The numerical summaries are provided in Table 5, from which we can see Eclipse takes 0.75 seconds while our simulator costs 0.278 seconds.

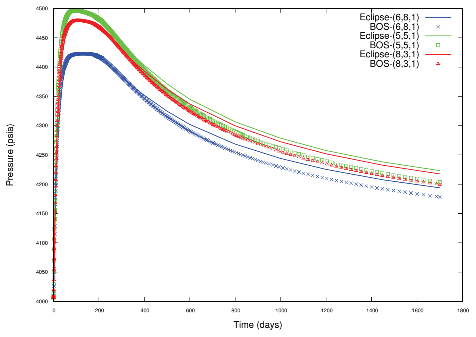


Fig. 7. Pressure profiles at selected positions.

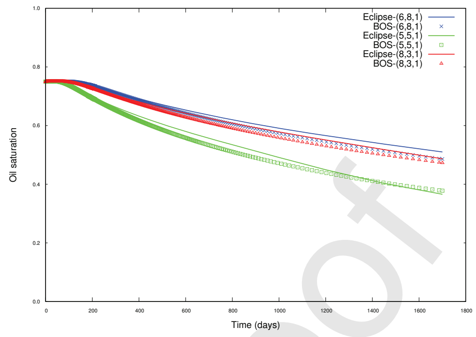


Fig. 8. Oil saturation profiles at selected positions.

Table 5. Numerical summaries for the validation case.

	Steps	# Newton	Avg N.	# Solver	Avg L.	Time (s)
BOS	12	29	2.42	84	2.90	0.278
Eclipse	27	69	2.56	295	4.28	0.75

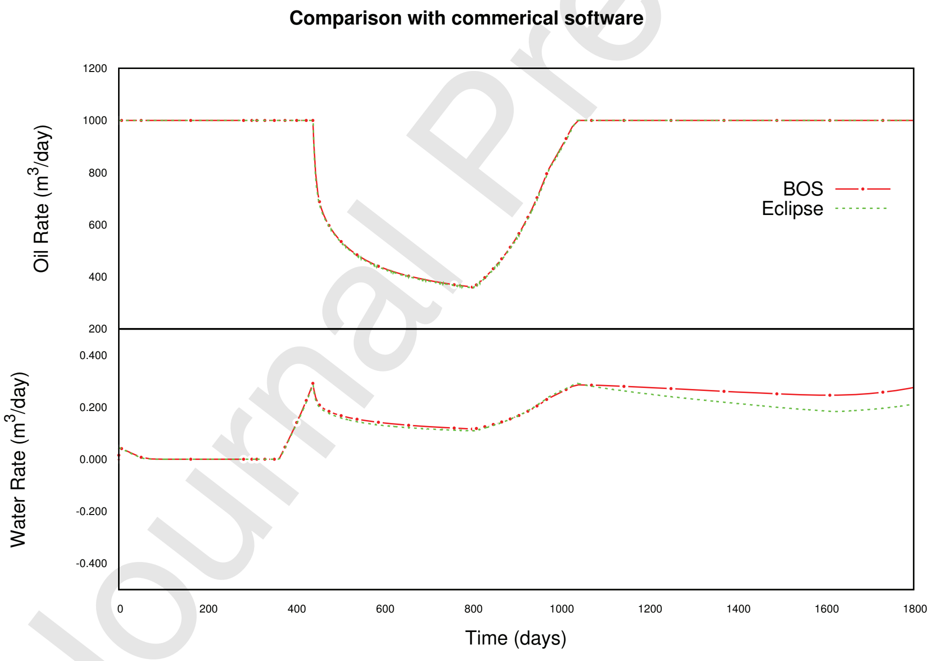


Fig. 9. Production profile comparison between BOS and Eclipse with multiple layer wells and capillary gravity equilibrium state.

As the results shown, our simulator matches well with Eclipse both on the well operation curves and pressure/oil saturation profiles.

4.2. Numerical Performance

In this section, a famous geological model, SPE10 [36], is applied to study the performance of numerical methods empolyed to the parallel simulator. The SPE10 has highly heterogeneous porosity and permeabilities, which introduce numerical difficulties to non-linear methods and linear solvers. The model has a grid dimension of $60 \times 220 \times 85$ and 1.1 million gird blocks. Five wells exist in this model, one injection well in the center and four wells in four corners.

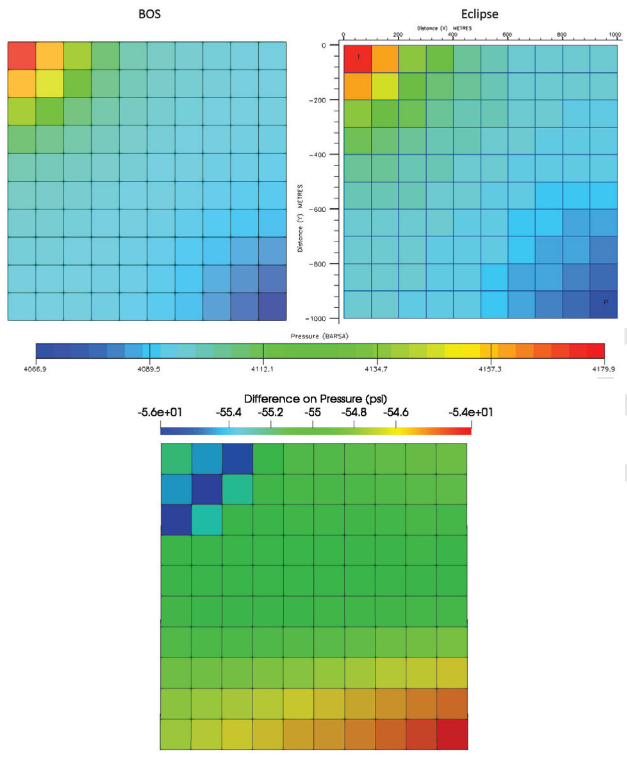


Fig. 10. Pressure profile at the shallowest layer for 3D model after 1800 days operation.

The simulation period is 200 days, and the maximal time step is 15 days. The linear systems have around 3.3 million unknowns. Different preconditioners and decoupling methods are studied using 16 CPU cores and 16 MPIs.

4.2.1. Newton Method

The model has a grid dimension of $95 \times 192 \times 5$, and five wells. The simulation period is 5480 days and maximal time step is 100 days. The linear solver is BICGSTAB with a tolerance of 5e-3 and maximal iterations of 40, and the preconditioner is CPR-FPF. The termination tolerences of Newton method are changed to investigate its sensitivity on numerical performance. The model is small and up to 8 CPU cores (one computation node) are employed. Numerical summaries are shown in Table 6, 7, 8 and 9. These tables show that the number of MPI processes (CPU cores) affect the numerical performance of the polymer flooding simulator. For example, in Table 6, linear iterations have big jump when more than 1 CPU cores are employed. The tables also show when smaller tolerance is applied, the simulation takes more Newton iterations, linear solver iterations and computing time to complete. Each time step requires more Newton iterations on average. From the results we can see the numerical methods effective. When tolerance is large, each time step requires around 4 Newton iterations, and when tolerance is very small, around 5 Newton iterations are required. In all cases, we can see that the linear solver and preconditioner are efficient, less than 7 iterations are required to solve each Jacobian system. Figure 12 presents speedup, which indicates that the simulator has similar speedup for different Newton termination tolerance.

4.2.2. Preconditioners

Four different preconditioners are studied, in which the standard Newton method is applied for non-linear system, the maximal Newton iterations are 10, its termination tolerance is 1e-4, and the linear solver is BICGSTAB with a tolerance of 1e-3 and maximal iterations of 100. GJE decoupling method is applied for linear systems. Numerical summaries are shown in Table 10. From the table, we can see that the CPR-like preconditioners have better convergence than the RAS preconditioner, which is also the slowest method. In all four preconditioners, the CPR-FPF preconditioner is the fastest.

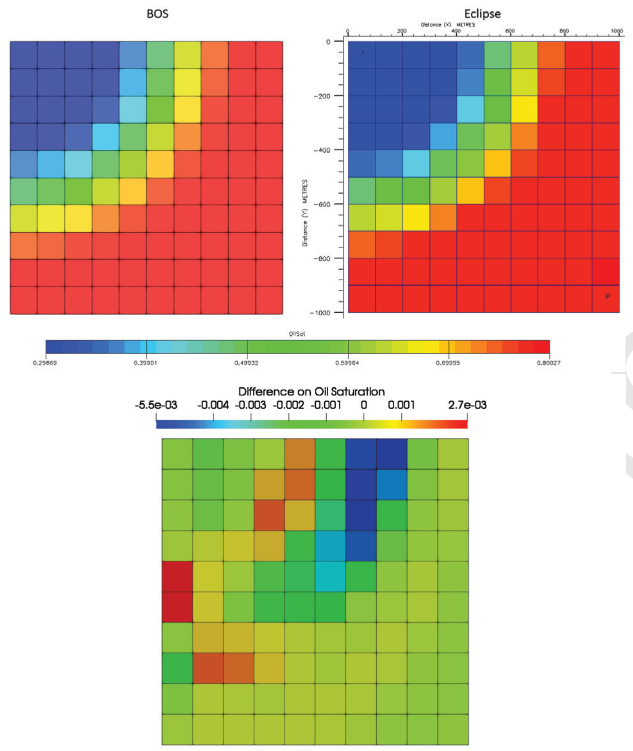


Fig. 11. Oil saturation profile at the shallowest layer for 3D model after 1800 days operation.

Table 6. Numerical summaries for tolerance 1e-2

CPU Cores	Steps	# Newton	Avg N.	# Solver	Avg L.	Time (s)
1	81	202	2.49	756	3.74	938.58
2	81	202	2.49	1124	5.56	596.85
4	81	202	2.49	1165	5.76	292.41
8	81	202	2.49	1246	6.16	180.20

Table 7. Numerical summaries for tolerance 1e-4

CPU Cores	Steps	# Newton	Avg N.	# Solver	Avg L.	Time (s)
1	81	284	3.51	1200	4.23	1393.18
2	81	261	3.22	1518	5.82	819.18
4	81	262	3.23	1563	5.97	392.02
8	81	255	3.15	1596	6.26	223.57

Table 8. Numerical summaries for tolerance 1e-6

CPU Cores	Steps	# Newton	Avg N.	# Solver	Avg L.	Time (s)
1	81	364	4.49	1598	4.39	1821.30
2	81	335	4.13	2009	5.99	1068.81
4	81	340	4.20	2119	6.23	518.87
8	81	337	4.16	2217	6.57	304.89

4.2.3. Decoupling Methods

This section studies the numerical performance of decoupling methods, and five methods are applied to the SPE10 projects. The standard Newton method is applied, whose tolerance and maximal iterations are 1e-4 and 10. The linear

Table 9. Numerical summaries for tolerance 1e-8

CPU Cores	Steps	# Newton	Avg N.	# Solver	Avg L.	Time (s)
1	81	456	5.63	2051	4.49	2318.13
2	81	395	4.87	2429	6.15	1221.11
4	81	398	4.92	2498	6.28	611.84
8	81	392	4.84	2590	6.61	389.34

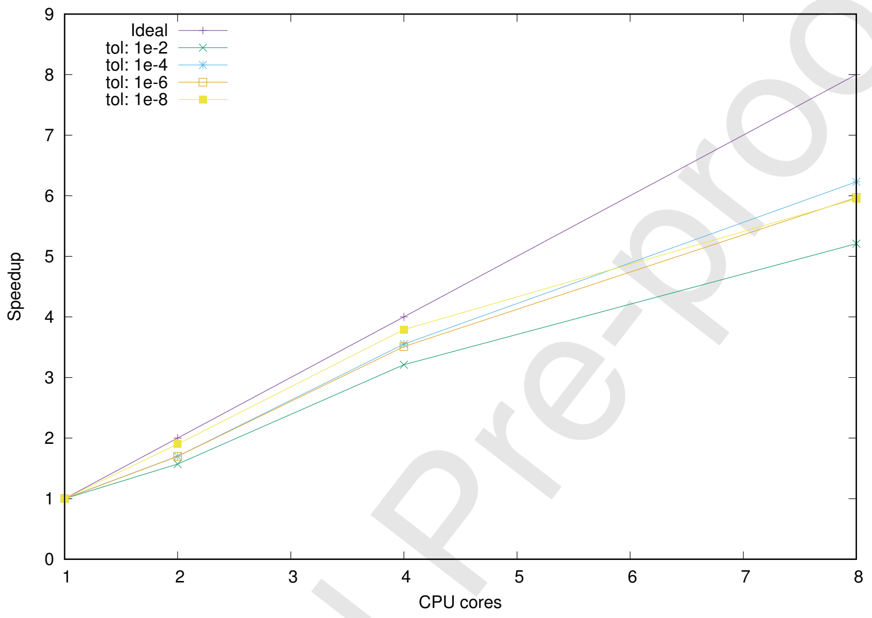


Fig. 12. Speedup using difference tolerance and CPU cores.

Table 10. Numerical summaries of different preconditioners

Method	Steps	# Newton	Avg N.	# Solver	Avg L.	Time (s)
CPR	164	478	2.91	4570	9.56	2893.68
CPR-FP	165	537	3.25	3787	7.05	2809.59
CPR-FPF	162	465	2.87	3599	7.74	2715.52
RAS	160	449	2.81	19641	43.74	3441.90

solver is BICGSTAB, its tolerance is 1e-3, the maximal iterations are 100, and preconditioner is CPR-FPF method. Numerical results are shown in Table 11. From the table, we can see that the GJE method and the ABF have good convergence, which are good decoupling method for polymer flooding simulations. The FRS method fails to converge. However, when the FRS method is applied as the first stage, the convergence of the GJE method and the ABF method is improved. Since matrix multiplication is involved for the ABF method, its performance is not as good as the GJE method.

Table 11. Numerical summaries of different decoupling methods

Method	Steps	# Newton	Avg N.	# Solver	Avg L.	Time (s)
GJE	162	465	2.87	3599	7.74	2715.52
ABF	161	446	2.77	3585	8.04	3175.67
FRS	NA	NA	NA	NA	NA	NA
FRS-GJE	161	451	2.80	3485	7.73	2855.46
FRS-ABF	161	442	2.75	3539	8.01	3386.82

4.3. Scalability

In this section, the performance of our simulator is reported in terms of speedup. The speedup, s, is defined as the ratio of the elapsed time when executing a simulation by r cores to the execution time when n processors are used:

$$s_n = \frac{T_r}{T_n} \tag{41}$$

In the ideal scenario, the ideal speedup is n/r, and the amount of time reduces linearly with the number of CPUs rising. From a performance point of view, our simulator is demonstrated to provide significant speedup with respect to multi-core CPUs through our tests.

4.3.1. Low resolution reservoir model

The first performance test is carried out by IBM BladeCenter HS23 high performance blade server on a polymer flooding project with $95 \times 192 \times 5$ grid blocks in the x, y and z directions, respectively. The reservoir properties, such as porosity and permeability, vary along different layers, as Table 12 shows. The initial reservoir pressure is 4000 psi at a reference depth of 6150 ft. The simulation was based on water flood by a vertical and horizontal well patten for 3980 $days$, followed by polymer flood for almost 360 $days$. Four vertical injection wells are distributed at the corners of the reservoir, while a horizontal producer is drilled at the center of the reservoir.

Table 12. Reservoir description

Layer	Porosity	Permeability (mD)		
		X	Y	Z
1	0.17393	326.4	980.6	163.4
2	0.1694	445.3	1335.8	222.6
3	0.25714	148.9	446.6	74.4
4	0.17344	118.8	356.4	59.4
5	0.1187	71.2	213.6	35.6

Numerical summaries are listed in Table 13 and Table 14, respectively for our simulator and Eclipse. The default setting in Eclipse is used to solve the linear equations by Orthomin and nested factorization preconditioner [20]. Same as Eclipse, our simulator uses Newton's method to solve the non-linear governing system. For this case, BICGSTAB and CPR-FPF preconditioner are combined to solve the linear system. As the Figure 1 shown, The computation processes include linearizing the equations, solving the linear equations and checking this linear solution being good enough for non-linear solution. The number of linear iterations is a measure of how efficiency of the preconditioner and linear solver work on the solution process.

The results reported in Figure 13 demonstrate the performance of our simulator. Simulations are performed by 4 (the reference) to 128 cores. The speedup profiles on the whole simulation and linear solver are represented by red and green lines in Figure 13. The blue line shows the speedup of Eclipse that runs up to 32 cores. As Figure 13 shown, our simulator keeps reducing the simulation time even with 128 cores are applied. However, the speedup of Eclipse deviates from ideal and it is not scalable anymore when the processor number exceeds 8 cores.

Table 13. Numerical summaries of different cores for low resolution reservoir model.

Cores	Steps	# Newton	Avg N.	# Solver	Avg L.	Time(s)
4	43	137	3.19	805	5.88	87.28
8	43	137	3.19	855	6.24	50.43
16	43	137	3.19	895	6.53	27.32
32	43	137	3.19	906	6.61	16.25
64	43	138	3.21	1008	7.30	10.06
128	43	141	3.28	1164	8.25	8.50

Table 14. Numerical summaries by Eclipse for low resolution reservoir model.

Cores	# Newton	# Solver	Avg L.	Time(s)
4	134	2056	15.34	48.06
8	134	2025	15.11	37.35
16	134	2045	15.26	59.07
32	133	2130	16.02	93.2

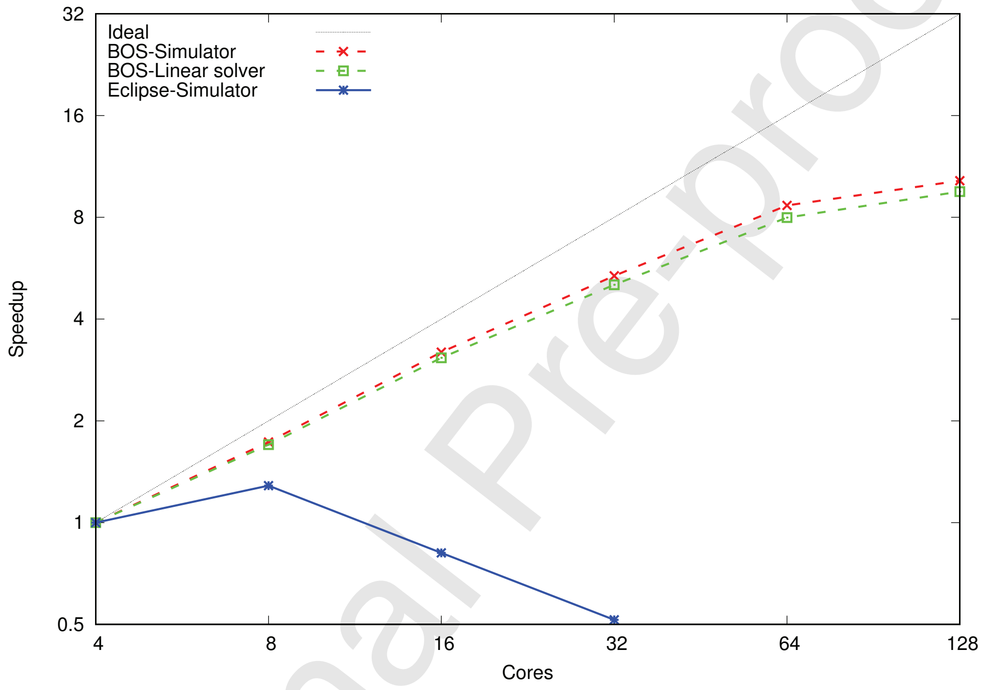


Fig. 13. Speedup of simulators and linear solver (reference is 4 cores).

4.3.2. High resolution reservoir model

The second performance test is carried out by Cedar cluster supplied by Compute Canada. The Cedar totally consists of 58,416 CPU cores and 584 GPU devices, spreaded across 1,542 nodes. Cedar is designed to support multiple simultaneous parallel jobs of up to 1024 broadwell cores or 1536 skylake cores in a fully non-blocking manner. For larger jobs the interconnect has a 2:1 blocking factor, i.e., even for jobs running on several thousand cores, Cdear provides a high-performance interconnect. Theoretical peak double precision performance of Cedar is 936 teraflops for each CPU. The same reservoir description as in the last case is used except with the refined grid size of $951 \times 1920 \times 15$. Simulations were performed from 128 (the reference) to 2048 cores.

This example tests the speedup of the platform as well as the solution of linear systems (including the BICGSTAB iteration solver and the CPR preconditioner). As Figure 14 shown, the model is very close to ideal behavior up to 512 cores. When more than 512 processors are used, although total CPU time keeps decreasing with more processors used, parallel performance of the simulator deviates more from ideal behavior. To better understand this, the time cost of the simulator is divided into two parts, which are spent to solve and prepare the linear system. It can be observed that the section that influences most of the total CPU time is the cost to solve the linear system. Table 15 summarizes the numerical performance from 128 to 2048 cores. This example also shows the capability of the simulator who can deal with more accurate reservoir models at multi-million and even multi-billion grid blocks under accepted run time if there are enough computation resources available.

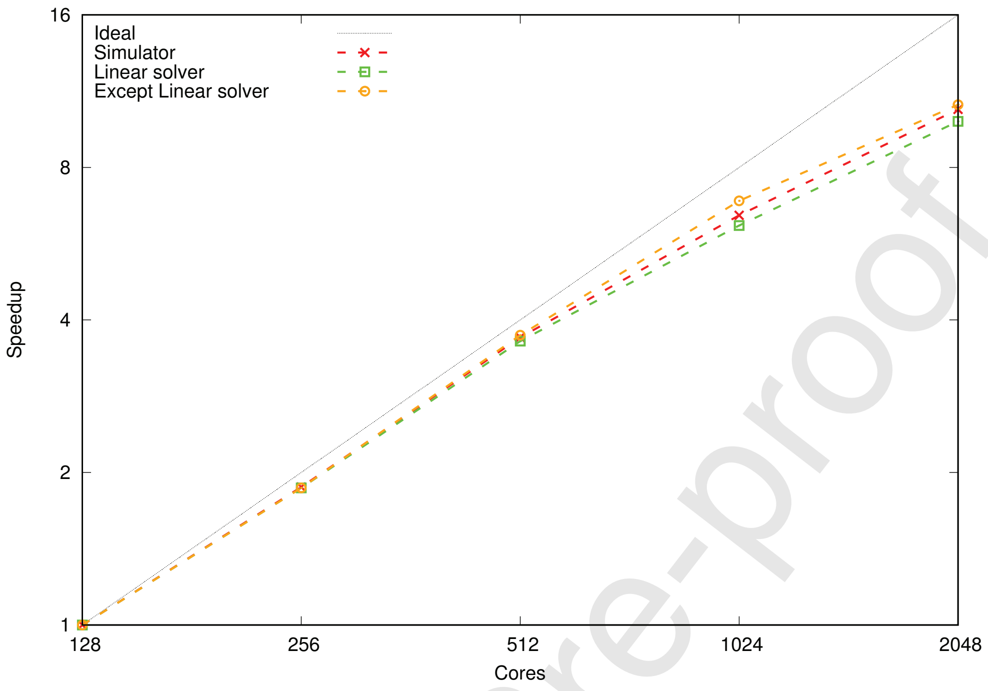


Fig. 14. Speedup of simulator and linear solver on 27.4×10^6 grid blocks (reference is 128 cores).

Table 15. Numerical summaries of different cores for high resolution reservoir model.

Cores	Steps	# Newton	Avg N.	# Solver	Avg L.	Time(s)
128	175	1401	8.01	4580	3.27	16988.0
256	177	1426	8.06	4681	3.28	9073.6
512	175	1392	7.95	4587	3.30	4597.8
1024	175	1396	7.98	4850	3.47	2640.2
2048	175	1395	7.97	5954	4.27	1631.6

5. Conclusions

The results of this work demonstrate that coupled fluid flow and polymer adsorption phenomena can be effectively simulated and distributed among multi-core CPUs. High performance computing techniques are applied through our in-house simulator, which has the capability to build and solve large-scale reservoir systems, especially reservoir models with high resolutions and complexity. The rapid advancements in parallel computer hardware and software technologies are adapted to improve the efficiency of polymer flooding processes. A satisfactory parallel efficiency of our simulator is demonstrated for a complex application with up to 27 million grid blocks by 2048 cores.

Acknowledgments

The support of Department of Chemical and Petroleum Engineering, University of Calgary and Reservoir Simulation Group is gratefully acknowledged. The research is partly supported by NSERC/Energi Simulation Chair, Alberta Innovate Chair, 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).

[1] D. Coumou, S. K. Matthäi, S. Geiger, T. Driesner, A parallel FE-FV scheme to solve fluid flow in complex geologic media, Computers and Geosciences 34 (2008) 1697–1707.

[2] A. H. Dogru, Megacell Reservoir Simulation, Journal of Petroleum Technology 52 (2000).

[3] D. A. Edwards, D. Gunasekera, J. Morris, G. Shaw, K. Shaw, P. A. Fjerstad, J. Kikani, J. Franco, V. Hoang, L. Quettier, Reservoir Simulation : Keeping Pace with Oilfield Complexity, Oilfield Review 23 (2012) 4–15.

[4] D. J. Pye, Improved Secondary Recovery by Control of Water Mobility, Journal of Petroleum Technology 16 (1964) 911–916.

[5] B. Sandiford, Laboratory and Field Studies of Water Floods Using Polymer Solutions to Increase Oil Recoveries, Journal of Petroleum Technology 16 (1964) 917–922.

[6] L. W. Lake, R. L. Schmidt, P. B. Venuto, A Niche for Enhanced Oil Recovery in the 1990s., Oilfield Review 17 (1992) 62–67.

[7] D. Wang, R. S. Seright, Z. Shao, J. Wang, Key Aspects of Project Design for Polymer Flooding at the Daqing Oil Field, SPE Reservoir Evaluation & Engineering 11 (2008) 1117–1124.

[8] H. Saboorian-Jooybari, M. Dejam, Z. Chen, Heavy oil polymer flooding from laboratory core floods to pilot tests and field applications: Half-century studies, Journal of Petroleum Science and Engineering 142 (2016) 85–100.

[9] Z. Chen, Y. Ma, G. Chen, A sequential numerical chemical compositional simulator, Transport in Porous Media 68 (2007) 389–411.

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

[11] P. Bondor, G. Hirasaki, M. Tham, Mathematical Simulation of Polymer Flooding in Complex Reservoirs, Society of Petroleum Engineers Journal 12 (1972) 369–382.

[12] P. M. Lutchmansingh, Development and application of a highly implicit, multidimensional polymer injection simulator, Ph.D. thesis, The Pennsylvania State University, 1987.

[13] J. H. Abou-Kassem, M. E. Osman, A. M. Zaid, Architecture of a multipurpose simulator, Journal of Petroleum Science and Engineering 16 (1996) 221–235.

[14] Y.-B. Chang, G. A. Pope, K. Sepehrnoori, A higher-order finite-difference compositional simulator, Journal of Petroleum Science and Engineering 5 (1990) 35–50.

[15] C. Han, M. Delshad, K. Sepehrnoori, G. A. Pope, A Fully Implicit, Parallel, Compositional Chemical Flooding Simulator, SPE Journal 12 (2007).

[16] N. F. Najafabadi, M. Delshad, C. Han, K. Sepehrnoori, Formulations for a three-phase, fully implicit, parallel, EOS compositional surfactant-polymer flooding simulator, Journal of Petroleum Science and Engineering 86-87 (2012) 257–271.

[17] L. Patacchini, R. de Loubens, A. Moncorgé, A. Trouillaud, Four-Fluid-Phase, Fully Implicit Simulation of Surfactant Flooding, SPE Reservoir Evaluation & Engineering 17 (2014) 271–285.

[18] M. Behzadinasab, E. A. Grein, K. Sepehrnoori, Development of a parallel chemical flooding reservoir simulator, International Journal of Oil, Gas and Coal Technology 16 (2017).

[19] Z. Chen, Reservoir Simulation Mathematical Techniques in oil recovery, Society for Industrial and Applied Mathematics, Philadelphia, PA, 2007.

[20] Schlumberger, Eclipse Technical Description, 2014.

[21] CMG, IMEX User's Guide, 2011.

[22] D. W. Green, G. Willhite, Enhanced Oil Recovery, Society of Petroleum Engineers, 1998.

[23] K. A. Lie, H. M. Nilsen, A. Rasmussen, X. Raynaud, An unconditionally stable splitting method using reordering for simulating polymer injection, in: ECMOR 2012 - 13th European Conference on the Mathematics of Oil Recovery, 2012.

[24] F. A. Dumkwu, A. W. Islam, E. S. Carlson, Review of well models and assessment of their impacts on numerical reservoir simulation performance, Journal of Petroleum Science and Engineering 82-83 (2012) 174–186.

[25] D. W. Peaceman, Interpretation of Well-Block Pressures in Numerical Reservoir Simulation(includes associated paper 6988), Society of Petroleum Engineers Journal 18 (1978) 183–194.

[26] D. W. Peaceman, Interpretation of Well-Block Pressures in Numerical Reservoir Simulation With Nonsquare Grid Blocks and Anisotropic Permeability, Society of Petroleum Engineers Journal 23 (1983).

[27] C. L. Palagi, K. Aziz, Modeling Vertical and Horizontal Wells With Voronoi Grid, SPE Reservoir Engineering 9 (1994) 15–21.

[28] L. S. K. Fung, H. Su, J. C. T. Tan, K. Hemanthkumar, J. A. Pita, A Fully-Implicit Fully-Coupled Well Model for Parallel Mega-Cell Reservoir Simulation, in: SPE Technical Symposium of Saudi Arabia Section, 14-16 May, Dharan, Saudi Arabia, 2005.

[29] K. H. Coats, In-Situ Combustion Model, SPE Society of Petroleum Engineers of AIME 20 (1980) 533–554.

[30] H. Liu, K. Wang, B. Yang, M. Yang, R. He, L. Shen, H. Zhong, Z. Chen, Dynamic Load Balancing Using Hilbert Space-Filling Curves for Parallel Reservoir Simulations, in: SPE Reservoir Simulation Conference, 20-22 February, Montgomery, Texas, USA, 2017.

[31] H. Liu, K. Wang, Z. Chen, K. E. Jordan, J. Luo, H. Deng, A Parallel Framework for Reservoir Simulators on Distributed-memory Supercomputers, in: SPE/IATMI Asia Pacific Oil & Gas Conference and Exhibition, 20-22 October, Nusa Dua, Bali, Indonesia, 2015.

[32] H. Liu, K. Wang, Z. Chen, A family of constrained pressure residual preconditioners for parallel reservoir simulations, Numerical Linear algebra with applications 23 (2016) 120–146.

[33] S. Lacroix, Y. V. Vassilevski, M. F. Wheeler, Decoupling preconditioners in the implicit parallel accurate reservoir simulator (IPARS), Numerical Linear Algebra with Applications 8 (2001) 537–549.

[34] R. Bank, T. Chan, The alternate-block-factorization procedure for systems of partial differential equations, BIT Numerical … 4 (1989) 938–954.

[35] R. Scheichl, R. Masson, J. Wendebourg, Decoupling and block preconditioning for sedimentary basin simulations, Computational Geosciences 7 (2003) 295–318.

[36] M. Christie, M. Blunt, Tenth SPE Comparative Solution Project: A Comparison of Upscaling Techniques, SPE Reservoir Evaluation & Engineering 4 (2001) 308–317.

Declaration of interests

☒ The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

☐The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: