Comput Geosci (2017) 21:1189–1202
DOI 10.1007/s10596-017-9683-7

ORIGINAL PAPER

A parallel framework for a multipoint flux mixed finite element equation of state compositional flow simulator

Benjamin Ganis[1] · **Gurpreet Singh**[1] · **Mary F. Wheeler**[1]

Received: 6 September 2016 / Accepted: 29 June 2017 / Published online: 20 July 2017

Abstract Mathematical models of physical problems are becoming increasingly complex and computationally intensive. At the same time, computing hardware is becoming more parallelized with an increasing number of cores promoting simultaneous tasks. In this work, we present a parallel, equation of state (EOS), compositional flow simulator for evaluating CO_2 sequestration, enhanced oil recovery techniques such as gas flooding, and other subsurface porous media applications. Using the multipoint flux mixed finite element (MFMFE) method for spatial discretization, it can handle complex reservoir geometries using general distorted hexahedral grid elements, as well as satisfy local mass conservation and compute accurate phase fluxes. A parallel framework for the MFMFE is presented that has been extended to the highly non-linear, EOS, compositional flow model. Much of the non-linearity is due to the local flash and stability calculations associated with interphase mass transfer and phase behavior. Parallel multigrid linear solver libraries such as HYPRE are utilized to solve the algebraic problems on each Newton step. We perform a variety of strong and weak parallel scaling studies up to 10 million elements and 1024 processors, and discuss possible load balancing issues.

✉ Benjamin Ganis
bganis@ices.utexas.edu

Gurpreet Singh
gurpreet@ices.utexas.edu

Mary F. Wheeler
mfw@ices.utexas.edu

[1] Center for Subsurface Modeling, The Institute for Computational Engineering and Sciences (ICES), The University of Texas at Austin, Austin, TX, USA

Keywords Equation of state · Compositional flow · Multipoint flux · Mixed finite element method · General hexahedral grids · High performance computing

1 Introduction

Compositional flow models are used for a wide range of problems in flow and transport in porous media such as CO_2 sequestration, gas or chemical injection for enhanced oil recovery, contaminant plume migration, and ground water remediation. Field-scale simulations are now commonly used to develop field strategies and are often computationally intensive. The spatial and temporal scale of the problem as well as model complexity further exacerbates this issue requiring a large number of degrees of freedom to be evaluated. Reduction of computational time is necessary in order to allow multiple scenario evaluation and hence decision making in a time constrained manner. Fortunately, the cost of computational resources is reducing day by day making it more accessible. Advances in parallel architecture necessitate the development of new parallel algorithms to obtain increased performance with new computing hardware. Moreover, these developments must be thoroughly evaluated using scalability studies for efficiency. Of particular interest here is the non-linear, local calculations associated with the equation of state (EOS), phase behavior model. These local calculations can often lead to load balancing issues and consequently loss of parallel scalability.

Many solution schemes exist for compositional flow modeling, following the early developments by Watts et al. [23], Coats et al. [3], Acs et al. [2], and others. There are many technical differences in these solution algorithms, as these schemes may use implicit or explicit methods, complementarity conditions, Jacobian approximations, or other solution techniques. One of the first numerical solution schemes for solving a compositional flow model on early serial computing platforms was due to [6]. Since that time, some efforts to parallelize compositional flow models to modern distributed memory high performance computing architectures include [5, 14, 16, 19, 22]. Note that each parallel framework is built upon a specific mathematical formulation and discretization scheme, which determine the sparsity of the linear systems and parallel communication requirements. In the present day, parallel computing with compositional flow models and related multi-physics applications is a very active area of research; the literature contains many studies on pushing the limits of large scale simulations [4], the scalability of linear solvers and preconditioners [10], accelerating the convergence of nonlinear solvers [13], improving discretization techniques [11], and applications to new computing technologies [15].

The discretization scheme we employ in this work is known as the multipoint flux mixed finite element (MFMFE) scheme [26]. The MFMFE method is a finite element interpretation of the multipoint flux approximation (MPFA) methods [1, 7] from the finite volume community. On structured hexahedral grids, this method has four flux degrees of freedom per face in three dimensions and a single scalar degree of freedom per element [12]. This higher order scheme is known to be convergent on distorted hexahedral grids, whereas typical two-point flux schemes commonly used in reservoir simulation are not. It supports full tensor permeability, and the method is locally mass conservative, which is a critical property for modeling accurate transport as required by multiphase flow with compressibility, capillarity, and gravity. Although it is classified as a mixed finite element method, it employs special quadrature rules to locally eliminate flux degrees of freedom around every vertex in terms of the surrounding pressure degrees of freedom, which gives a positive definite linear system with a 27-point stencil and circumvents the need to solve a saddle-point system. The MFMFE spatial discretization scheme was recently extended to an equation of state compositional flow model formulation in [21]. In this work, we present a parallel framework for the this model, which leverages distributed computations using a message passing interface (MPI) library.[1]

The format of our paper is as follows. In Section 2, we give a brief description of the compositional model formulation restricting the discussion to details pertinent for adequately describing the parallelization task performed as a part of this work. In Section 3, we present the nonlinear algebraic equations associated with the hydrocarbon phase behavior model. In Section 4, we describe the linearized fully discrete form of the partial differential equations associated with the compositional model formulation. In Section 5, we describe the parallel framework used in this work, based upon a data decomposition technique for computational load allocation per processor along with MPI for communication between several processors. Finally, in Section 6, we present parallel scalability results, both strong and weak scaling, considering multiple numerical experiments differing in computational costs primarily due to local, non-linear, phase behavior calculations. The results identify load balancing issues and algorithmic bottlenecks arising due to these local calculations and provide insights for future dynamic domain decomposition techniques to address them. We give conclusions in Section 7.

2 Compositional model formulation

In this section, we present a compositional flow model formulation for component conservation in a multiphase, compressible flow along with constraints, initial, and boundary conditions. Several important assumptions that we make in our model include isothermal reservoir conditions, rock-fluid interactions are neglected, non-reactive flow, and a slightly-compressible aqueous phase exists as a pure water component.

2.1 Component conservation equations

The mole (or mass) conservation equation of component i for a multiphase flow system is given by,

$$\frac{\partial}{\partial t}\left(\sum_{\alpha}\phi S_\alpha \rho_\alpha \xi_{i\alpha}\right) + \nabla\cdot\sum_{\alpha}\left(\rho_\alpha\xi_{i\alpha}u_\alpha - \phi S_\alpha D_{i\alpha}\cdot\nabla\left(\rho_\alpha\xi_{i\alpha}\right)\right) = \sum_{\alpha} q_{i\alpha}, \quad \text{in } \Omega\times(0,T]. \tag{1}$$

Here, S_α is the saturation of phase α (ratio of volume of phase α to pore volume), ϕ the porosity (ratio of pore volume to bulk volume), $q_{i\alpha}$ the rate of injection of component i in phase α (mole, mass or volume basis), u_α the Darcy flux, $\xi_{i\alpha}$ the normalized mole (or mass) fraction of component i in phase α, and $D_{i\alpha}$ the diffusion-dispersion tensor of component i in phase α. Further, u_α is the phase flux given by the Darcy's law as,

$$u_\alpha = -K\frac{k_{r\alpha}}{\mu_\alpha}\left(\nabla p_\alpha - \rho_{m,\alpha} g\right). \tag{2}$$

Here, K is the absolute permeability of the porous rock matrix, $k_{r\alpha}$ the relative permeability, μ_α the viscosity, and $\rho_{m,\alpha}$ the mass density of phase α. Let N_i, q_i, F_i, and

[1]MVAPICH web page: http://mvapich.cse.ohio-state.edu.

J_i be the concentration, phase summed source/sink term, advective and diffusive fluxes of component i, respectively defined as,

$$N_i = \sum_\alpha \rho_\alpha S_\alpha \xi_{i\alpha}, \tag{3}$$

$$q_i = \sum_\alpha q_{i\alpha}, \tag{4}$$

$$F_i = -K \sum_\alpha \rho_\alpha \xi_{i\alpha} \frac{k_{r\alpha}}{\mu_\alpha} \left(\nabla p_\alpha - \rho_{m,\alpha} g\right), \tag{5}$$

and

$$J_i = -\left(\sum_\alpha \phi S_\alpha D_{i\alpha} \left(\nabla \rho_\alpha \xi_{i\alpha}\right)\right). \tag{6}$$

The component conservation equations can then be written in a concise form as,

$$\frac{\partial}{\partial t}(\phi N_i) + \nabla \cdot (F_i + J_i) = q_i, \quad \text{in } \Omega \times (0, T]. \tag{7}$$

2.2 Constraint, initial, and boundary conditions

The phase saturations are normalized volume fractions such that,

$$\sum_\alpha S_\alpha = 1. \tag{8}$$

Further, phase pressures (p_α) can be defined in terms of a reference phase pressure (p_{ref}) using the phase capillary pressure constraint as,

$$p_{c\alpha}(S_{ref}) = p_\alpha - p_{\text{ref}}. \tag{9}$$

Here, S_{ref} is the reference phase saturation. The reference phase is usually chosen to be the wetting phase pressure, however a different choice is also possible by altering the functional form of the capillary pressure function ($p_{c\alpha}(S_\alpha)$). For the ease of understanding, the reader might consider the porous medium to be water wet and therefore water as the reference phase.

We assume the reference phase pressure (p_{ref}) and component concentrations (N_i) as the primary unknowns where N_1 is the water component concentration, $\vec{N}_{HC} = N_{2,\ldots,N_c}$ the hydrocarbon component concentration vector and $\vec{N} = N_{1,\ldots,N_c}$ the component concentration vector. The phase saturations S_α can be calculated as functions of these primary unknowns as,

$$\begin{aligned} S_w(p_{ref}, N_1) &= \frac{N_w}{\rho_w}, \\ S_o(p_{ref}, \vec{N}_{HC}) &= \frac{(1-\nu)}{\rho_o} \sum_{i=2}^{N_c} N_i, \\ S_g(p_{ref}, \vec{N}_{HC}) &= \frac{\nu}{\rho_g} \sum_{i=2}^{N_c} N_i. \end{aligned} \tag{10}$$

Here, ν is the normalized mole fraction of the hydrocarbon gas phase, and o, w, and g represent the hydrocarbon oil, water and hydrocarbon gas phases, respectively. The saturation constraint (8) and capillary pressure (9) can be expressed in terms of these primary unknowns as,

$$\frac{N_w}{\rho_w} + \left(\frac{1-\nu}{\rho_o} + \frac{\nu}{\rho_g}\right) \sum_{i=2}^{N_c} N_i = 1, \tag{11}$$

and

$$p_{c\alpha}(p_{ref}, \vec{N}) = p_{c\alpha}(S_{ref}), \tag{12}$$

respectively. A slightly compressible (13a) and a cubic equation of state (13b) are used for water and hydrocarbon phases, respectively.

$$\rho_w(p_{ref}) = \rho_{w,0} exp\left[C_w(p_{\text{ref}} + p_{cw} - p_{\text{ref},0})\right] \tag{13a}$$

$$\rho_\alpha(p_{ref}, \vec{N}_{HC}) = \frac{p_\alpha}{Z_\alpha RT}, \alpha \neq \text{w} \tag{13b}$$

Here, ρ_α is the molar density of phase α and ρ_w the water phase density. The phase compressibility Z_α is evaluated using the Peng-Robinson equation of state [17]. We restrict ourselves to no flow boundary condition for the ease of model description such that,

$$(F_i + J_i) \cdot n = 0, \quad \text{on } \partial\Omega. \tag{14}$$

An initial condition is prescribed for the reference pressure and concentration as,

$$p_{\text{ref}} = p^0, \tag{15a}$$

and

$$N_i = N_i^0, \tag{15b}$$

respectively. The rock matrix may also be considered slightly compressible with the porosity $\phi = \phi_0(1 + c_r * (p_{ref} - p_0))$ as a function of reference pressure p_{ref} with constants initial porosity ϕ_0, rock compressibility c_r, and initial pressure p_0.

3 Hydrocarbon phase behavior model

The hydrocarbon phase partitioning $\vec{K}^{par}$, normalized mole fraction ν is calculated using an appropriate phase behavior model which relies upon a local equilibrium assumption. This local equilibrium assumption, also known as the isofugacity criteria, is described by a system of non-linear, algebraic equations,

$$g_i(p_{ref}, \vec{N}_{HC}, \vec{K}^{par}, \nu) = ln(\Phi_{io}) - ln(\Phi_{ig}) - ln K_i^{\text{par}} = 0 \quad i = 2, \ldots., N_c. \tag{16}$$

where $\Phi_{i\alpha}$ is the fugacity coefficient of component i in phase α defined by

$$
\begin{aligned}
ln(\Phi_{i\alpha}) &= -C_i + \frac{B_i}{B_\alpha}(\bar{Z}_\alpha - 1) - ln(\bar{Z}_\alpha - B_\alpha) \\
&\quad - \frac{A_\alpha}{2\sqrt{2}B_\alpha}\left(\frac{2\sum_{j=2}^{N_c}\xi_{j\alpha}A_{ij}}{A_\alpha} - \frac{B_i}{B_\alpha}\right) \\
&\quad \times ln\left(\frac{\bar{Z}_\alpha + (1+\sqrt{2})B_\alpha}{\bar{Z}_\alpha + (1-\sqrt{2})B_\alpha}\right). \qquad (17)
\end{aligned}
$$

The constants in this expression can be found in [21].

Further, a total mole/mass balance over all the components and phases is given by the Rachford-Rice (RR) [18] equation,

$$
f(\vec{N}_{HC}, \vec{K}^{par}, \nu) = \sum_{i=2}^{N_c} \frac{(K_i^{\text{par}} - 1)z_i}{1 + (K_i^{\text{par}} - 1)\nu} = 0. \qquad (18)
$$

Here, $\vec{z}$ is the overall component mole fraction and is related to the primary variable $\vec{N}_{HC}$ as,

$$
z_i(\vec{N}_{HC}) = \frac{N_i}{\sum_{i=2}^{N_c} N_i}. \qquad (19)
$$

The reader is referred to [21], and the references therein, for more details regarding the phase behavior model. The non-linear system of algebraic equations associated with the local, phase behavior model (16)–(18) results in addition of the phase partitioning coefficient $\vec{K}^{par}$ and normalized phase mole fraction ν to the existing set of primary unknowns (p_{ref} and $\vec{N}$). In practice, this system is solved with a local Newton iteration on every element.

4 Linearized fully discrete formulation

In this section, we present the system of linear algebraic equations obtained after Newton linearization of the non-linear, fully discrete compositional flow model equations. An implicit backward Euler scheme is used for the temporal discretization and an MFMFE scheme is used for the spatial discretization scheme. On a given time step, our compositional solution algorithm is sequential implicit as explained in the remainder of this section. For full details of the MFMFE scheme, the reader should consult [12, 21, 26]. For the convenience of description below, we assume that the diffusive flux J_i to be zero. The linearized, fully discrete form of the component flux (5) and conservation (7) equations is given by,

$$
\begin{aligned}
&\left\langle \frac{1}{\Lambda_{i,h}(p_{ref}^k, \vec{N}^k)} K^{-1}\delta F_{i,h}, v_h \right\rangle_{Q,E} \\
&- \left(\delta p_{\text{ref},h}, \nabla\cdot v_h\right)_E = -R_{1i}^k, \qquad (20)
\end{aligned}
$$

and

$$
\begin{aligned}
&\left(\frac{\phi_h(p_{ref}^k)}{\Delta t}\delta N_{i,h}, w_h\right)_E + \left(\frac{c_r\phi_0 N_{i,h}^k}{\Delta t}\delta p_{\text{ref},h}, w_h\right)_E \\
&\qquad + \left(\nabla\cdot\delta F_{i,h}, w_h\right)_E = -R_{2i}^k. \qquad (21)
\end{aligned}
$$

respectively. The $(\cdot,\cdot)$ denotes the usual L^2-inner product and $\langle\cdot,\cdot\rangle_Q$ denotes a special quadrature allowing local velocity elimination around element vertices. On distorted hexahedral grids, there are both symmetric and nonsymmetric quadrature rules for the MFMFE method given in [24]. Here, the notation $\delta\vec{x}$ is the Newton increment of a primary unknown $\vec{x}$ given by,

$$
\delta x = \vec{x}^{k+1} - \vec{x}^k. \qquad (22)
$$

The local mass matrix and right hand side for component i can be written as,

$$
\begin{pmatrix} A_i^k & B & 0 \\ B^T & C_i^k & D_i^k \end{pmatrix}
\begin{pmatrix} \delta F_i \\ \delta p_{\text{ref}} \\ \delta N_i \end{pmatrix}
= \begin{pmatrix} -R_{1i}^k \\ -R_{2i}^k \end{pmatrix}. \qquad (23)
$$

Here, δF_i, δp_{ref}, and δN_i are global unknown vectors of sizes equal to 4 (or 2) $\times$ number of edges of elements open to flow for the flux unknowns in 3D (or 2D) and equal to number of elements for pressure and concentrations unknowns. The first and second terms in Eq. 20 contribute to matrices A_i and B in Eq. 23, whereas the third term in Eq. 21 adds to matrix B^T. Note that the matrix B remains constant throughout the simulation and therefore the superscript for Newton iteration k is omitted. Further, D_i and C_i are diagonal matrices corresponding to the first and second terms in Eq. 21. Please note that the off-diagonal contributions to D_i due to the third term in Eq. 21 are neglected resulting in an approximate Jacobian construction. Eliminating δF_i in favor of cell centered quantities δp_{ref} and δN_i, the global linear system is given by,

$$
\left(C_i^k - B^T(A_i^{-1})^k B\right)\delta p_{ref} + D_i^k\delta N_i = -R_{2i}^k + B^T(A_i^{-1})^k R_{1i}^k \qquad (24)
$$

Similarly, a local linear system is constructed using the local phase behavior equations and saturation constraint. The saturation constraint, iso-fugacity criteria (16), and

Rachford-Rice equation (18) can be linearized in terms of the unknowns p_{ref}, N_i, K_i^{par}, and ν as,

$$\sum_\alpha \left(\frac{\partial S_\alpha}{\partial p_{\text{ref}}}\right)^k \delta p_{\text{ref}} + \sum_\alpha \sum_i \left(\frac{\partial S_\alpha}{\partial N_i}\right)^k \delta N_i + \sum_\alpha \sum_i \left(\frac{\partial S_\alpha}{\partial ln K_i^{\text{par}}}\right) \delta ln K_i^{\text{par}} + \sum_\alpha \left(\frac{\partial S_\alpha}{\partial \nu}\right)^k \delta \nu = 1 - \sum_\alpha S_\alpha^k = -R_{1,E}^k, \tag{25}$$

$$\Phi_{i\alpha} = \Phi_{i\alpha}(p_{\text{ref}}, \xi_{i\alpha}) = \Phi_{i\alpha}(p_{\text{ref}}, z_i, K_i^{\text{par}}) = \Phi_{i\alpha}(p_{\text{ref}}, N_i, K_i^{\text{par}}), \tag{26}$$

$$\left(\frac{\partial ln \Phi_{io}}{\partial p_{\text{ref}}}\right)^k \delta p_{\text{ref}} + \sum_{j=2}^{N_c} \left(\frac{\partial ln \Phi_{io}}{\partial N_j}\right)^k \delta N_j + \sum_{j=2}^{N_c} \left(\frac{\partial ln \Phi_{io}}{\partial ln K_j^{\text{par}}}\right)^k \delta ln K_j^{\text{par}} + \left(\frac{\partial ln \Phi_{io}}{\partial \nu}\right)^k \delta \nu - \left(\frac{\partial ln \Phi_{ig}}{\partial p_{\text{ref}}}\right)^k \delta p_{\text{ref}} - \sum_{j=2}^{N_c} \left(\frac{\partial ln \Phi_{ig}}{\partial N_j}\right)^k \delta N_j - \sum_{j=2}^{N_c} \left(\frac{\partial ln \Phi_{ig}}{\partial ln K_j^{\text{par}}}\right)^k \delta ln K_j^{\text{par}} - \left(\frac{\partial ln \Phi_{ig}}{\partial \nu}\right)^k \delta \nu - \sum_{j=2}^{N_c} \left(\frac{\partial ln K_i^{\text{par}}}{\partial ln K_j^{\text{par}}}\right)^k \delta ln K_j^{\text{par}} = -R_{2i,E}^k. \tag{27}$$

$$\sum_{i=2}^{Nc} K_{i,E}^{par} \left(\frac{z_{i,E}}{1 + (K_{i,E}^{par} - 1)\nu_E} - \frac{(K_{i,E}^{par} - 1) z_{i,E} \nu_E}{\left[1 + (K_{i,E}^{par} - 1)\right]^2} \right) \delta ln K_{i,E}^{par} - \frac{(K_{i,E}^{par} - 1)^2 z_{i,E}}{\left[1 + (K_{i,E}^{par} - 1)\right]^2} \delta \nu_E + \sum_{i=2}^{Nc} \left(\frac{1}{\sum_{j=2}^{Nc} N_{j,E}} - \frac{N_i}{\left(\sum_{j=2}^{Nc} N_{j,E}\right)^2} \right) \delta N_{i,E} = -R_{3,E}^k \tag{28}$$

The above equations can also be written in the matrix form as,

$$\begin{pmatrix} E_E^k & F_E^k & G_E^k & H_E^k \\ I_E^k & J_E^k & K_E^k & L_E^k \\ 0 & N_E^k & O_E^k & P_E^k \end{pmatrix} \begin{pmatrix} \delta p_{ref,E} \\ \delta N_E \\ \delta ln K_E^{\text{par}} \\ \delta \nu_E \end{pmatrix} = \begin{pmatrix} -R_{1,E}^k \\ -R_{2,E}^k \\ -R_{3,E}^k \end{pmatrix}. \tag{29}$$

Here, the δN_E and $\delta ln K_E^{par}$ are the local, unknown vectors of size $N_c - 2$ corresponding to an element E; $\delta N_E = \left(\delta N_{2,E}, \cdots, \delta N_{N_c,E}\right)^T$ and $\delta ln K_E^{par} = \left(\delta ln K_{2,E}^{par}, \cdots, \delta n K_{N_c,E}^{par}\right)^T$. The values of phase compressibilities (Z_α) are evaluated explicitly given pressure p_{ref}, temperature T, and component concentrations N_is. The subscript E is used to denote calculations on an element due to the local nature of the non-linear algebraic equations associated with phase behavior calculations as well as the saturation constraint. Eliminating $\delta ln K_E^{\text{par}}$ and $\delta \nu_E$ from the local linear system (29), we obtain another local linear system in cell-centered unknowns $\delta p_{ref,E}$ and δN_E,

$$\tilde{M}_E^k \delta p_{ref,E} + \tilde{N}_E^k \delta N_{ref,E} = -R_{4,E}^k \tag{30}$$

The local linear system (30) is then used to eliminated δN_i from the global linear system (24) to obtain a further reduced implicit system in the pressure unknown only. Once the pressure increments (δp_{ref}) are evaluated, the concentration increments (δN_i) are obtained via back substitution. For these reasons, the solution scheme is considered a sequential implicit scheme. Furthermore, the reduced linear system in the pressure unknown has the same sparsity pattern as the single-phase slightly compressible flow model. Therefore, no special pre-conditioners for the reduced linear system are required as in the case of a fully-implicit linear system, where oftentimes specialized two-stage preconditioners are applied to reduce computational costs during the linear solve [9].

5 Parallel framework

The parallelization of our compositional MFMFE model is similar to way that previous MFMFE flow models were parallelized in the Implicit Parallel Accurate Reservoir Simulator (IPARS) framework. Other models include the single and two-phase slightly compressible flow models [24, 25]. We use the MVAPICH MPI library for performing parallel distributed memory computations on high performance computing platforms. The basic strategy is known as data decomposition, and is its correct implementation is a highly nontrivial task. First, the problem is decomposed such at each processor owns a small piece of the domain, along with the necessary data and state variables. An example decomposition is shown in Fig. 1. Before each model computation step, all state variables require `update` routines using parallel communication, in order to fill ghost cells (sometimes called a halo) with all neighboring information. Particular care needs to be taken in well model computations to communicate shared well properties across all

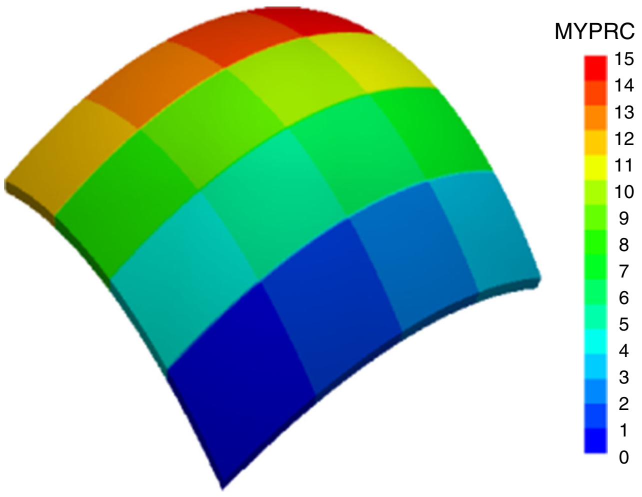


Fig. 1 Data decomposition of dome-shaped geometry into 16 processors

processors that own well elements. This ensures the problem and solution remain the same irrespective of the number of processors utilized at runtime. After all `update` calls are completed, the linear system can be assembled in parallel without communication, where each processor fills the rows of the Jacobian matrix and the entries of the residual vector corresponding to the elements that it owns. Next, the code interfaces with a parallel iterative solver library to solve the linear system, using parallel communication to perform dot product, matrix-vector product, and preconditioner application steps. During the entire process, whenever global norms are computed, for example to determine Newton convergence or time step stability calculations, `allreduce` operations are performed.

5.1 Parallel communication

The IPARS framework assumes each domain consists of a logically structured grid of either Cartesian bricks or distorted general hexahedra with non-planar faces. Capabilities also exist for multiblock simulations consisting of several domains, but are not investigated in this work. Some elements can be marked as inactive in a `keyout` array to conform to special geometric considerations. The philosophy is that structured grids have much less overhead than fully unstructured grids both in terms of model development and assembly time during computations. The hexahedral grids used in the MFMFE flow models are particularly well-suited to capture arbitrary reservoir surfaces and other geologic features such as fractures and faults, while maintaining the reduced overhead of a structured grid. As parallel communication is relatively more expensive than local computations, a minimal amount of arrays are called with `update` routines, and redundant calculations are performed in ghost elements whenever possible. The processor decomposition is performed aerially, with entire vertical columns of elements belonging to the same processor.

As discussed in Section 4, the MFMFE method on hexahedra reduces the saddle point system for the mixed finite element method to a positive-definite system for the cell-centered pressure variable, through the use of specialized quadrature rules to eliminate the velocity variables around every vertex [12]. This is analogous to the technique that was performed with the lowest-order Raviart-Thomas (RT0) finite element method on brick geometry, showing that finite difference methods are actually a special case of mixed methods with special quadrature [20]. In the RT0 case on bricks, the reduced system has a 7-point stencil for the pressure unknown, coupling each element with its neighbors in the cardinal directions. In the MFMFE case on hexahedra, the reduced system has a larger 27-point stencil for

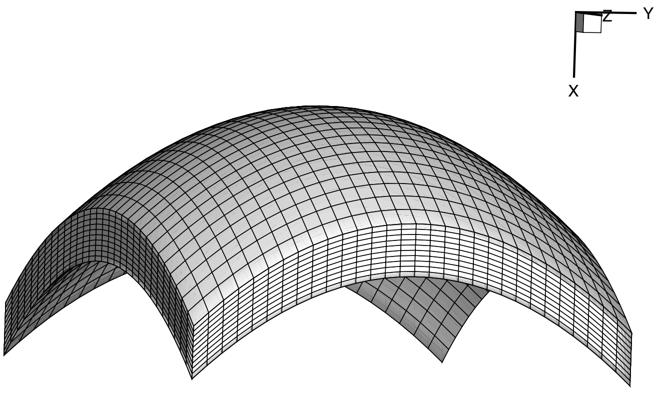
Table 1 Percentage of elements sent during `update` with 7-point and 27-point stencils for the strong scaling case with 1 million elements (top) and the weak scaling case (bottom)

Procs.	7-p update	27-p update
32	9.76%	10.6%
64	14.0%	15.1%
128	21.8%	24.5%
256	30.4%	35.4%
512	44.2%	53.5%
1024	62.8%	80.2%

Procs.	Elements	7-p update	27-p update
64	6.40e5	3.50%	4.24%
256	2.56e6	3.75%	4.54%
1024	1.02e7	3.88%	4.69%

Table 2 Solver parameters used in the numerical experiments in this work

`HYPRE` parameter	Value	
Relative linear tolerance	1.e-6	
Max. linear iterations	1000	
Max. Krylov subspace dimension	50	
Relaxation type	6	(Gauss-Seidel smoother)
Cycle type	1	(V-cycle)
Coarsening type	6	(Falgout coarsening)
Measure type	0	(local measure)
Number of sweeps	1	
Max. multigrid levels	20	
Truncation factor	0.	
Strong threshold	0.95	



```
xside=8500;                % Top depth at midpoints of sides
xtop=8000;                 % Top depth at center of domain
xd=5; yd=10; zd=10;        % Width grid blocks in each direction
nx=25; ny=200; nz=200;     % Number of elements in each direction

ymin=0; ymax=ny*yd;                       % Aerial extents
zmin=0; zmax=nz*zd;
ymid=(ymin+ymax)/2; zmid=(zmin+zmax)/2;   % Midpoint of domain

xhat = linspace(0,nx*xd,nx+1);            % Reference grid
yhat = linspace(ymin,ymax,ny+1);
zhat = linspace(zmin,zmax,nz+1);

A = [ ymin^2 ymin zmid^2 zmid 1; ...      % Set up linear system
      ymid^2 ymid zmin^2 zmin 1; ...
      ymid^2 ymid zmax^2 zmax 1; ...
      ymax^2 ymax zmid^2 zmid 1; ...
      ymid^2 ymid zmid^2 zmid 1 ];
b = [xside; xside; xside; xside; xtop];
c = A\b;                                  % Determine constants

for i=1:nx+1                              % Fill physical grid
    for j=1:ny+1
        for k=1:nz+1
            x(i,j,k) = c(1)*yhat(j)^2 + c(2)*yhat(j) + ...
                c(3)*zhat(k)^2 + c(4)*zhat(k) + c(5) + xhat(i);
            y(i,j,k) = yhat(j);
            z(i,j,k) = zhat(k);
        end
    end
end
```

Fig. 2 Dome-shaped geometry with a coarse structured grid of hexahedral elements (*left*), and example Matlab code for dome-shaped grid generation (*right*)

the pressure unknown, coupling each element with neighbors in both cardinal and all diagonal directions. The larger MFMFE stencil makes calls to `update` more expensive. In Table 1, we compare the number elements necessary to transfer between processors during an `update` with 7-point and 27-point stencils. It is evident that interprocess communication is more expensive for multipoint flux models compared to two-point flux approximations.

5.2 HYPRE multigrid solver

The parallel iterative linear solver that we utilize in this work is the `HYPRE` [8] library. As the linear systems arising from our multiphase flow discretization are non-symmetric and ill-conditioned, we employ the `GMRES` solver with algebraic multigrid preconditioning. Our experience with the parallelization of our model has guided us to arrive upon the following set of solver parameters for the simulations reported in this paper, see Table 2. Parallel scaling experiments with the `HYPRE` library have demonstrated that it can be an efficient solver for our model. We note here that our equation-of-state compositional model is a fully-implicit formulation with the primary unknowns chosen to be 1 reference pressure and $(N_c - 1)$ concentrations. With the solution to this system, you can determine all N_c unknown concentrations. In our sequential implicit formulation, concentrations contributing to off-diagonal entries are Newton lagged in order to form a reduced linear system with a single reference pressure unknown on every Newton step. Therefore, the use of algebraic multigrid preconditioning is sufficient to obtain fast convergence with the iterative solver.

Table 3 Component properties (top): critical temperatures, critical pressures, critical volumes, acentric factors, molecular weights, parachor, and volumetric shift. Binary interaction coefficients (bottom), unspecified means zero

Comp.	TC	PC	ZC	AC	MW	PR	VS
C1	343.0	667.8	.290	.0130	16.04	71.	0.
C3	665.7	616.3	.277	.1524	44.10	151.	0.
C6	913.4	436.9	.264	.3007	86.18	271.	0.
C10	1111.8	304.0	.257	.4885	142.29	431.	0.
C15	1270.0	200.0	.245	.6500	206.00	0.	0.
C20	1380.0	162.0	.235	.8500	282.00	0.	0.

	C1	C3	C15	C20
C1			.05	.05
C3			.005	.005
C15	.05	.005		
C20	.05	.005		

6 Results

In this section, we perform both strong and weak parallel scaling studies with the implementation of our equation of state compositional model. We vary the difficulty of the simulations through the phase behavior. Our simulations were run on the Stampede supercomputer at the Texas Advanced Computing Center.[2] Since we report timing results, we note that each compute node has dual 8-core Intel Xeon E5-2680 processors and 32 gigabytes of memory, connected with a

[2]Stampede web page: https://www.tacc.utexas.edu/stampede/.

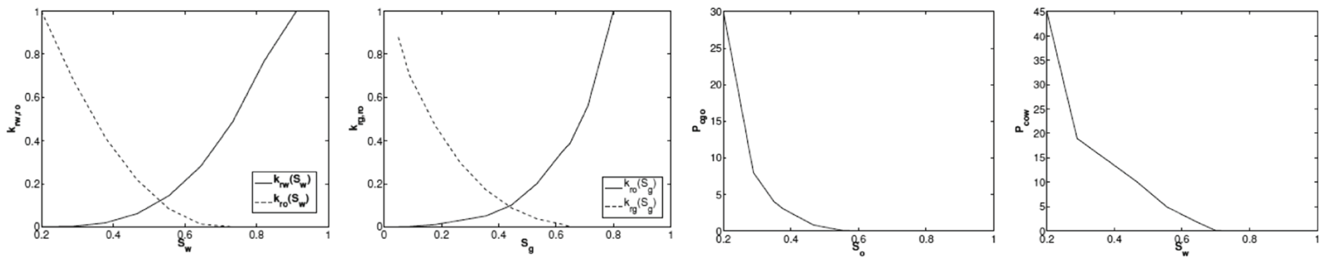


Fig. 3 From *left to right*: oil and water relative permeability versus s_w, oil and gas relative permeability versus s_g, gas-oil capillary pressure versus s_o, and oil-water capillary pressure versus s_w curves

high-speed infiniband network (except Example 5, which was run on a newer set of compute nodes). Runtimes are reported as the maximum number of seconds elapsed across all processors rounded to the nearest integer, and the asterisk next to total time denotes that trivial initialization and visualization times were subtracted. Unless otherwise specified, the processor decomposition consists of square patches of roughly the same number of elements, as shown in Fig. 1. Results with different numbers of processors were verified to produce the same solution within specified tolerances for the same examples.

Geometry To show that our model is capable of handling arbitrary geometries with general hexahedra, we chose a simple but nontrivial dome-shaped domain, illustrated in Fig. 2. Our grid is generated as follows. Assume the x-coordinate points downward in the direction of gravity. Our grid is defined as a mapping from a Cartesian reference grid in $(\hat{x}, \hat{y}, \hat{z})$–space onto a physical grid of hexahedra in (x, y, z)–space. The physical grid has the ansatz $x = c_1\hat{y}^2 + c_2\hat{y} + c_3\hat{z}^2 + c_4\hat{z} + c_5 + \hat{x}$, $y = \hat{y}$, and $z = \hat{z}$. To determine the constants $c_1, \ldots, c_5$, we evaluate the equation for x at five arbitrarily specified points on the top surface $\hat{x} = 0$ (the center point of the domain and the midpoints of each of the four sides), and then solve the resulting 5×5 linear system. This mapping is then used to generate the remaining points of the grid. For reproducibility, we give Matlab code for grid generation in Fig. 2.

Model parameters The simulation time is fixed at $T = 100$ [days]. The initial time step is $\Delta t = 0.01$ [days] with a multiplier of 1.05, a maximum of 1 [day], and maximum saturation change of 0.5 for time step control. There is a uniform porosity of $\phi = 0.3$ and no rock compressibility $c_r = 0$. Unless otherwise specified, there is a uniform permeability $K = 50$ [md]. The initial conditions are water saturation $s_w(0) = 0.2$, gas saturation $s_g(0) = 0.0$, and oil pressure $p_o(0) = 2000$ [psi]. There is no diffusion-dispersion tensor. Gravitational force is present. The water phase properties are compressibility $c_w = 3.3\text{e-}6$ [1/psi], viscosity $\mu_w = 0.7$ [cp], and reference pressure $p_{w,ref} = 0$ [psi]. Standard densities are $\rho_{w,ref} = 62.4$ [psi], $\rho_{o,ref} = 56.0$ [psi], and $\rho_{g,ref} = 0.04228$ [psi]. The composition of the reservoir has 6 components, named C1, C3, C6, C10, C15, and C20. Component parameters are summarized in Table 3. The isothermal reservoir temperature is 160 °C. There is one separator for calculating fluids in place with a pressure of 14.7 [psi] and temperature of 60 °C. The relative permeability and capillary pressure curves are given in Fig. 3.

6.1 Example 1: strong scaling, water flood

To demonstrate strong parallel scalability in the following examples, we use a fixed dome-shaped grid with $25 \times 200 \times 200 = 1$ million elements, and examine runtimes as the number of processors is increased. Note that as the number of processors is increased, the portion of the domain that is owned by one processor becomes smaller. The parameters for the grid are depths $x_{\text{top}} = 8000$ [ft] and $x_{\text{side}} = 8500$ [ft], and each grid element size is $5 \times 10 \times 10$ [ft^3]. There are five vertical wells in a five-spot pattern, with one producer at the center of the domain and one injector at the center of each of the four quadrants.

In this example, the water is injected into the four injection wells at a constant bottom hole pressure (BHP) of 4000 [psi]. The production well BHP is pressure specified at 2000 [psi]. The initial reservoir composition is $\{0.0, 0.0, 0.6, 0.0, 0.0, 0.4\}$, representing the non-aqueous mole fractions of the six components, respectively. This combination gives a two-phase water flood example with trivial flash calculations for phase behavior. Simulation results are shown in Fig. 4. The water phase displaces the oil phase, no gas phase appears, and the two present component concentrations decrease in proportional amounts.

A breakdown of runtimes for the water flood case is given in Table 4. The number of processors was doubled in six cases from 32 to 1024 processors.[3] When the number of

[3]To emphasize the importance and necessity of high performance parallel computations with respect to time and memory constraints, we also report that a single processor case took a runtime of 24 h to reach simulation time 35.53 days and required 20.6 gigabytes of memory.

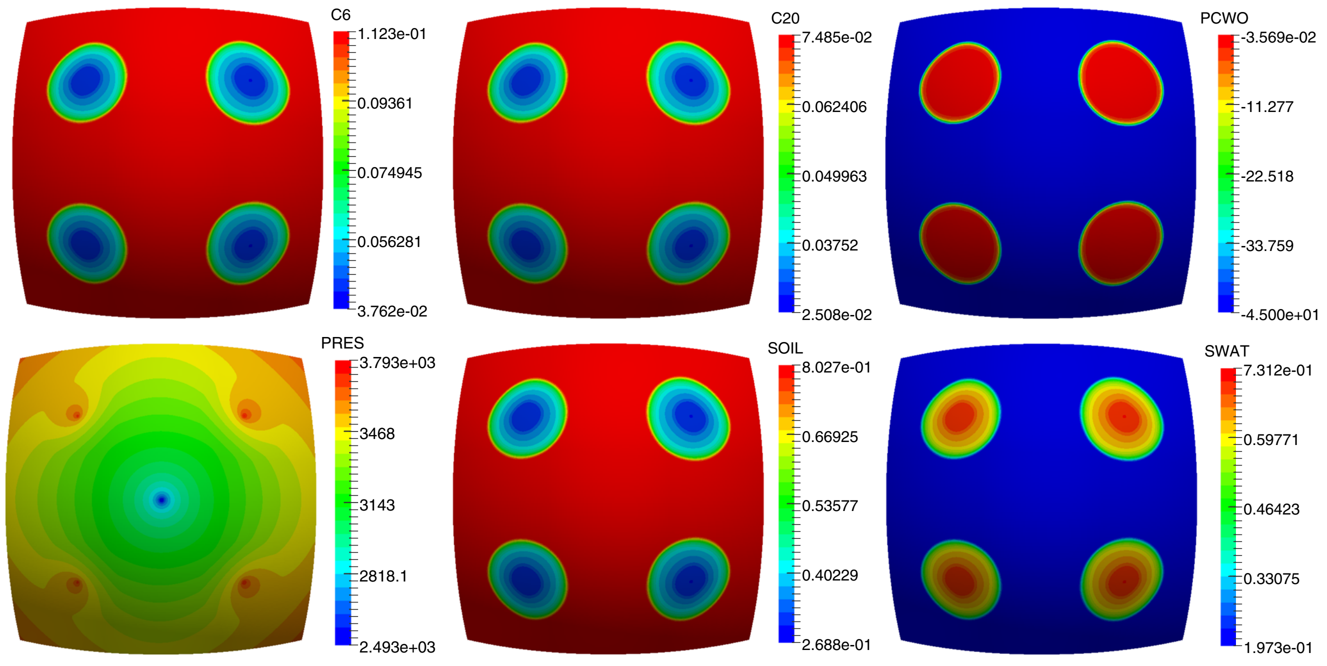


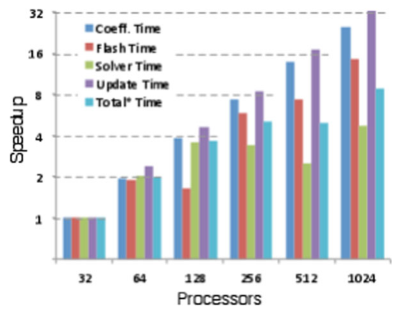
Fig. 4 Top view of simulation results at final time for Example 1. From *left to right*: C6 concentration, C20 concentration, water-oil capillary pressure, oil pressure, oil saturation, water saturation

processors are doubled, the optimal speedup is for runtimes to be reduced by half. Observe that the time spent assembling the coefficients for the linear system is nearly optimal, as this requires no communication, and scales with the number of elements per processor. The flash calculation is a sequential procedure with a loop over all the elements that a processor owns. It is a function of element composition, whereby more time is spent in elements with more difficult phase behavior. Depending on how the processor boundaries divide the domain, certain processors may have many more elements with complicated behavior than other processors. This may potentially cause load balancing issues in simulations where flash calculation time is a large portion of total time. In this case, flash scales well because calculations are trivial. The `HYPRE` solver we employ is an algebraic multigrid preconditioned GMRES iteration with parameters listed in Table 2. In strong scaling cases, there is always a point where the expense of interprocess communication offsets any gains in reducing computations. For this case, the solver begins to struggle at 256 processors. Although a further decrease in solver time happens at 1024 processors, speedup is far from optimal here. The number of cumulative linear solver iterations increased from 45,062 at 32 processors to 52,893 at 1024 processors. The number of cumulative Newton iterations remained constant at 2690 as did the number of time steps at 2689. The time spent in `update` scales optimally, because the the size of the messages become smaller and the parallel architecture can easily handle the quantity of messages simultaneously. The time spent in all of these tasks contributes to the overall total time. In this example, we see an overall decrease in total runtime all the way to 1024 processors.

Table 4 Breakdown of runtimes showing strong parallel scalability (left) and corresponding speedup factors normalized to 32 processors (right) in Example 1

Runtimes

Procs.	Coeff.	Flash	Solve	Update	Total*
32	4358	127	2755	288	7303
64	2235	67	1364	121	3722
128	1156	79	774	63	2033
256	590	22	821	34	1473
512	319	18	1116	17	1541
1024	178	9	585	8	909



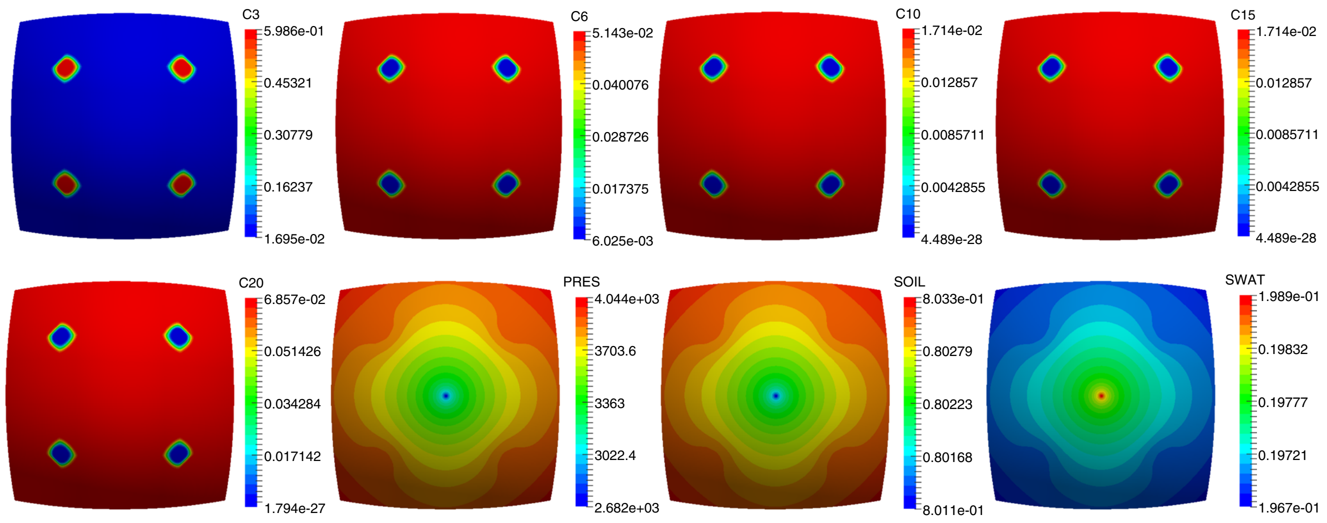


Fig. 5 Top view of simulation results at final time for Example 2. From *left to right*: C3 concentration, C6 concentration, C10 concentration, C15 concentration, C20 concentration, oil pressure, oil saturation, water saturation

6.2 Example 2: strong scaling, simple phase behavior

In this example, the four injectors are switched to gas injection at a constant bottom hole pressure (BHP) of 4000 [psi] with mole fraction composition {0.0, 0.0, 0.99, 0.01, 0.0, 0.0, 0.0}. The production well BHP remains pressure specified at 2000 [psi]. The initial reservoir mole fraction composition has been changed to {0.0, 0.1, 0.3, 0.1, 0.1, 0.4}. This combination gives a relatively simple phase behavior, resulting in nearly single phase behavior. Simulation results are shown in Fig. 5. The injected gas dissolves into the oil phase, and the increased pressure helps to displace the oil into the production well.

A breakdown of runtimes for the simple phase behavior case is given in Table 5, where the number of processors is again doubled six times from 32 to 1024. Both the coefficient assembly time and `update` time remain optimal as expected, demonstrating the parallel efficiency of the compositional model assembly and parallel framework. However, this time the flash calculation does not scale well, although it remains a small fraction of total time. This is because the phase behavior near the injection wells is somewhat more expensive than the rest of the domain. When this collection of elements happen to be divided into more processors, the flash time decreases, otherwise it stagnates. The linear solver is a large portion of total time, and it also scales worse in this case, never recovering after 256 processors. The number of cumulative linear solver iterations increased from 39,468 at 32 processors to 44,375 at 1024 processors. The number of cumulative Newton iterations remained constant at 3348 as did the number of time steps at 3347. The solver performance caused the overall runtime to stop decreasing after 256 processors.

6.3 Example 3: strong scaling, hard phase behavior

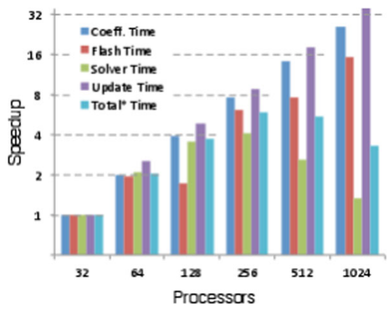
This example differs from the previous case, whereby the four gas injectors have their mole fraction composition changed to {0.0, 0.99, 0.01, 0.0, 0.0, 0.0, 0.0}. This leads to relatively hard phase behavior, resulting in a two-phase simulation. Simulation results are shown in Fig. 5. The injected gas exists in both oil and gas phases, pushing the fluids to the injection well in a more complex way (Fig. 6).

A breakdown of runtimes for the hard phase behavior case is given in Table 6. Only 64 to 1024 processors are reported, due to unexpected problems with 32 processors.

Table 5 Breakdown of runtimes showing strong parallel scalability (left) and corresponding speedup factors normalized to 32 processors (right) in Example 2

Runtimes

Procs.	Coeff.	Flash	Solve	Update	Total*
32	4602	66	2633	374	7619
64	2291	85	1270	147	3734
128	1191	97	747	77	2068
256	602	27	645	43	1302
512	328	22	1011	21	1416
1024	181	11	1983	10	2305



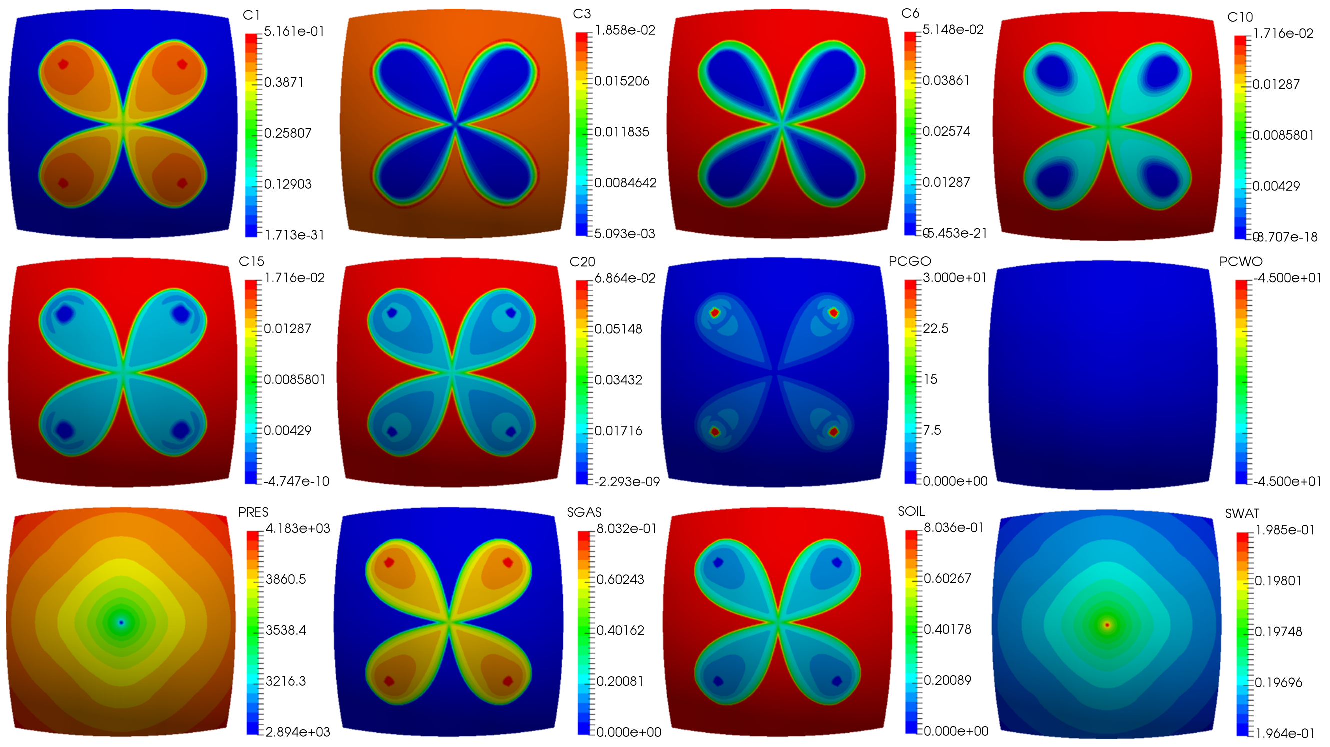


Fig. 6 Top view of simulation results at final time for Example 3. From *left to right*: C1 concentration, C3 concentration, C6 concentration, C10 concentration, C15 concentration, C20 concentration, gas-oil capillary pressure, water-oil capillary pressure, oil pressure, gas saturation, oil saturation, water saturation

We again confirm coefficient assembly time and `update` time are nearly optimal. We remark coefficient time is roughly twice as large as the previous two cases. Flash calculations are an order of magnitude more expensive in runtime compared to the previous case, although it scales well because they are less localized and hence more frequently divided by processor decomposition. The linear solver time scales to 256 processors, and again decreases slightly at 1024 processors. The number of cumulative linear solver iterations increased from 90,541 at 64 processors to 96,895 at 1024 processors. The number of cumulative Newton iterations remained constant at 10,885 as did the number of time steps at 6622. This combination of factors caused the total time to decrease all the way to 1024 processors.

6.4 Example 4: weak scaling, simple phase behavior

In this example, we repeat the simple phase behavior case using the same parameters as Example 2, but this time perform a weak scaling study. We assign each processor to own 10,000 elements, and perform runs with 64, 256, and 1024 processors. The size of the elements remain $5 \times 10 \times 10$ [ft^3], as do the grid depth parameters $x_{top} = 8500$ [ft] and $x_{side} = 8000$ [ft]. In this way, the dome-shaped reservoir becomes larger as the number of processors are increased, and the slope becomes more gradual. We chose to perform weak scaling in this way because constant element size will require roughly the same CFL criteria and lead to roughly the same time step size. (On the other hand, if we had chosen

Table 6 Breakdown of runtimes showing strong parallel scalability (left) and corresponding speedup factors normalized to 64 processors (right) in Example 3

Runtimes

Procs.	Coeff.	Flash	Solve	Update	Total*
64	7510	404	3208	470	11530
128	3845	317	1993	262	6426
256	1956	105	1405	138	3616
512	1064	72	3404	68	4750
1024	592	36	1598	34	2460

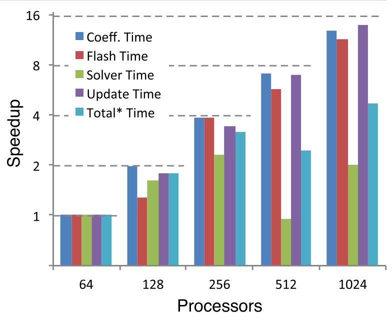


Table 7 Breakdown of runtimes showing weak parallel scalability in Example 4

Runtimes						
Procs.	Elements	Coeff.	Flash	Solve	Update	Total
64	640,000	295	14	255	17	576
256	2,560,000	176	9	342	10	550
1024	10,240,000	172	8	440	9	732

to keep reservoir size fixed and decrease element size as processors increased, then time step size would necessarily also decrease.) The well locations are also changed, keeping the producer in the center, and the injectors at the centers of the four quadrants. We admit the problem and solution are different on the three refinement levels, but convey that each one has the same basic characteristics and are thus comparable. We remark that strong scaling studies are difficult for highly nonlinear transient models such as our equation of state compositional model. The optimal behavior for weak scaling studies is for the times to remain constant as both the size of the problem and number of processors are increased concurrently.

A breakdown of runtimes for the weak scaling case is given in Table 7. Coefficient assembly, flash, and update times actually improved from 64 to 256 processors, but then remained roughly constant from 256 to 1024 processors. The linear solver time steadily increased each time processors were increased. Surprisingly, the number of linear solver iterations actually decreased from 86,361 at 64 processors to 35,940 at 1024 processors. This could be attributed to the fact that both problem and solution are changed with number of processors, so it is not a true weak scaling study. Number of cumulative Newton steps decrease from 6362 at 64 processors to 3504 at 1024 processors. Number of time steps decrease from 6361 at 64 processors to 3503 at 1024 processors. In spite of the linear solver increase, the overall total runtime remains roughly constant from 64 to 256 processors and increases quite modestly at 1024 processors, showing a positive result for weak scaling with our model.

6.5 Example 5: strong scaling, water flood, heterogeneous permeability

In the final example, we repeat the water flood in Example 1 in a strong scaling study, but this time include highly heterogeneous permeability, shown in Fig. 7. We also note here that this example was run on a newer set of compute nodes than the previous examples, on the stampede2 supercomputer with Intel Xeon Phi 7250 KNL processors (68 cores per node, 4 hardware threads per core) and 96 GB DDR4 RAM per node. We utilized 32 cores per node in this strong scaling study.

Each of the 20 layers of the permeability K is an independent realization of a stochastic permeability field on the domain $(0, 2000)^2$ of the form $K = exp(Y)$, where $Y(\mathbf{x}, \omega) = E[Y](\mathbf{x}) + Y'(\mathbf{x}, \omega)$. A Karhunen-Loève (KL) expansion for the mean-removed log permeability $Y'(\mathbf{x}, \omega)$ is first computed from the specified covariance function evaluated as a series expansion

$$C_Y(\mathbf{x}, \bar{\mathbf{x}}) = \sigma_Y^2 \exp\left[\frac{-|x_1 - \bar{x}_1|}{\eta_1} - \frac{|x_2 - \bar{x}_2|}{\eta_2}\right]$$
$$= \sum_{j=1}^{\infty} l_j f_j(\mathbf{x}) f_j(\bar{\mathbf{x}}).$$

The parameters used for this test are correlation lengths $\eta_1 = 17.0$, $\eta_2 = 30.0$, expectation $E[Y] = 2.1$, and variance $\sigma_Y = 5.0$. The series for the KL expansion was next truncated after N=400 terms

$$Y'(\mathbf{x}, \omega) \approx \sum_{j=1}^{N} \xi_j(\omega) \sqrt{l_j} f_j(\mathbf{x}),$$

where ξ_j are normal random variables with zero mean and unit variance. For the procedure for computing the eigenvalues l_j and eigenfunctions f_j of this series, the interested

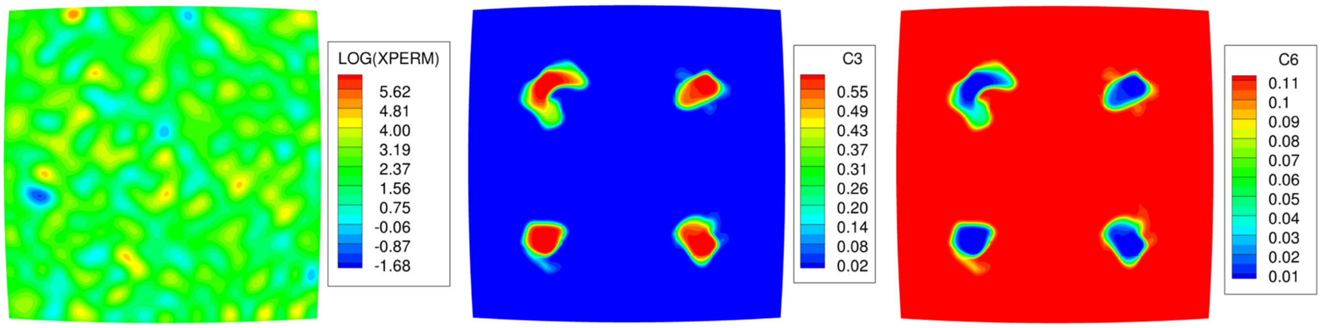
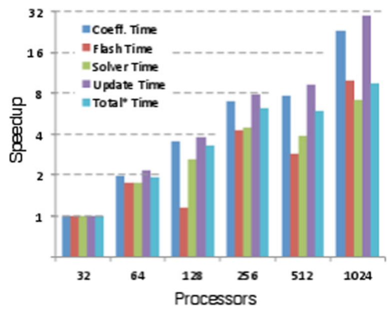


Fig. 7 Top view of Example 5. From *left to right*: logarithm of x-permeability component, C3 concentration at final time, C6 concentration at final time

Table 8 Breakdown of runtimes showing strong parallel scalability (left) and corresponding speedup factors normalized to 32 processors (right) in Example 5

Runtimes

Procs.	Coeff.	Flash	Solve	Update	Total*
32	37678	661	14417	2134	53122
64	18972	378	8157	991	27602
128	10559	574	5509	567	15976
256	5362	155	3184	271	8535
512	4883	229	3746	228	8873
1024	1637	65	2007	70	5619



reader can consult Appendix A in [27]. Finally, note that due to the geometric mapping from reference elements to the distorted hexahedral physical elements in the multipoint flux mixed finite element method's quadrature rules [24], the scalar heterogeneous permeability becomes a full tensor on the physical grid.

A breakdown of runtimes for the heterogeneous water flood case is given in Table 8. The modification of the permeability from a homogeneous $K = 50$ [md] to the heterogeneous field described above resulted in numerical experiments being six to eight times more expensive. We reiterate that Example 5 was run on newer hardware than Example 1, due to the replacement of compute nodes during the revision of this article. Nevertheless, the overall behavior of the parallel scalability is similar to the homogeneous water flood case in Example 1. Moreover, the proportion of each component of the runtime (coefficient, flash, solve, update) compared to the total runtime is also similar in behavior to the homogeneous water flood case in Example 1.

7 Conclusions

In this work, we have described in detail the work that was performed to parallelize our equation of state compositional multipoint flux mixed finite element reservoir simulator. Numerical results with distorted hexahedral grids have demonstrated that we are able to obtain favorable strong and weak parallel scaling up to 1024 processors and 10 million elements. We see that interprocess communication with the MFMFE model is relatively more expensive than two-point flux approximation models. Both coefficient setup time and variable updates show almost perfect parallel scaling in our implementation. As the performance of the linear solver is oftentimes a significant portion of the total runtime, it is important to interface with a good parallel solver library with a preconditioner that has been calibrated to the model. The HYPRE library that we used performed adequately for our model, although there is always room for improvement in every solver.

Some aspects of the compositional model such as flash calculations may lead to load balancing issues, but they are not always expensive or unbalanced. In this work, we have observed that coefficient and solver times dominated flash calculation times, but this may not always be the case. Under certain circumstances, it may be possible for flash calculations to become expensive in certain areas of the domain, especially when the fluid composition is near a critical point. In this case, processors which are largely made up of simple phase behavior would block until processors with more difficult behavior complete their calculations. Future work will include the implementation of load-balancing strategies such as the computation of error indicators and the use of specialized hardware for flash calculations.

Acknowledgements This work was supported by DOE grant FG02-04ER25617, DOE NETL grant DE-FE0023314, NSF grant 1546553, Statoil grant UTA13-000884, Saudi Aramco grant UTA11-000320, and the CSM industrial affiliates program. The authors acknowledge the Texas Advanced Computing Center (TACC) at The University of Texas at Austin for providing HPC resources that have contributed to the research results reported within this paper.

References

1. Aavatsmark, I., Barkve, T., Bøe, O., Mannseth, T.: Discretization On unstructured grids for inhomogeneous, anisotropic media. Part I: Derivation of the methods. SIAM J. Sci. Comput. **19**(5), 1700–1716 (1998)
2. Acs, G., Doleschall, S., Farkas, E., et al.: General purpose compositional model. Soc. Pet. Eng. J. **25**(04), 543–553 (1985)
3. Coats, K.H., et al.: An equation of state compositional model. Soc. Pet. Eng. J. **20**(05), 363–376 (1980)
4. Dogru, A.H., Fung, L.S.K., Middya, U., Al-Shaalan, T.M., Pita, J.A., HemanthKumar, K., Su, H.J., Tan, J.C.T., Hoy, H., Dreiman, W.T., et al.: A next-generation parallel reservoir simulator for giant reservoirs. In: SPE/EAGE Reservoir Characterization Andamp; Simulation Conference (2009)
5. Dogru, A.H., Sunaidi, H.A., Fung, L.S., Habiballah, W.A., Al-Zamel, N., Li, K.G., et al.: A parallel reservoir simulator for large-scale reservoir simulation. SPE Reserv. Eval. Eng. **5**(01), 11–23 (2002)
6. Douglas, J. Jr.: The numerical solution of a compositional model in petroleum reservoir engineering. SIAM-AMS Proc. **2**, 54–59 (1970)

7. Edwards, M.G., Rogers, C.F.: Finite volume discretization with imposed flux continuity for the general tensor pressure equation. Comput. Geosci. **2**(4), 259–290 (1998)
8. Falgout, R., Ulrike, Y.: HYPRE:A Library of high performance preconditioners. Comput. Sci., ICCS **2002**, 632–641 (2002)
9. Ganis, B., Kumar, K., Pencheva, G., Wheeler, M.F., Yotov, I., et al.: A multiscale mortar method and two-stage preconditioner for multiphase flow using a global jacobian approach. In: SPE Large Scale Computing and Big Data Challenges in Reservoir Simulation Conference and Exhibition. Society of Petroleum Engineers. SPE-172990-MS (2014)
10. Gries, S., Stüben, K., Brown, G.L., Chen, D., Collins, D.A., et al.: Preconditioning for efficiently applying algebraic multigrid in fully implicit reservoir simulations. SPE J. **19**(04), 726–736 (2014)
11. Hajibeygi, H., Tchelepi, H.A., et al.: Compositional multiscale finite-volume formulation. SPE J. **19**(02), 316–326 (2014)
12. Ingram, R., Wheeler, M.F., Yotov, I.: A Multipoint Flux Mixed Finite Element Method on Hexahedra. SIAM J. Numer. Anal. **48**(4), 1281–1312 (2010)
13. Jenny, P., Tchelepi, H.A., Lee, S.H.: Unconditionally convergent nonlinear solver for hyperbolic conservation laws with s-shaped flux functions. J. Comput. Phys. **228**(20), 7497–7512 (2009)
14. Killough, J.E., Bhogeswara, R., et al.: Simulation of compositional reservoir phenomena on a distributed-memory parallel computer. J. Petrol. Tech. **43**(11), 1–368 (1991)
15. Naumov, M., Arsaev, M., Castonguay, P., Cohen, J., Demouth, J., Eaton, J., Layton, S., Markovskiy, N., Reguly, I., Sakharnykh, N., et al.: Amgx: A library for gpu accelerated algebraic multigrid and preconditioned iterative methods. SIAM J. Sci. Comput. **37**(5), S602–S626 (2015)
16. Parashar, M., Wheeler, J.A., Pope, G., Wang, K., Wang, P., et al.: A new generation eos compositional reservoir simulator: Part ii-framework and multiprocessing. SPE Reservoir Simulation Symposium (1997)
17. Peng, D.-Y., Robinson, D.B.: A new two-constant equation of state. Ind. Eng. Chem. Fundam. **15**(1), 59–64 (1976)
18. Rachford, H.H., Rice, J.D.: Procedure for use of electronic digital computers in calculating flash vaporization hydrocarbon equilibrium. Trans. Am. Inst. Min. Metall. Eng. **195**, 327–328 (1952)
19. Reme, H., Øye, G.Å., Espedal, M.S., Fladmark, G.E.: Parallelization of a compositional reservoir simulator. In: Numerical Treatment of Multiphase Flows in Porous Media, pp. 244–266. Springer (2000)
20. Russell, T.F., Wheeler, M.F.: Finite element and finite difference methods for continuous flows in porous media. Math. Reservoir Simul. **1**, 35–106 (1983)
21. Singh, G., Wheeler, M.F.: Compositional flow modeling using a multipoint flux mixed finite element method. Comput. Geosci. **20**, 421–435 (2015)
22. Wang, P., Yotov, I., Wheeler, M.F., Arbogast, T., Dawson, C., Parashar, M., Sepehrnoori, K.: A new generation EOS compositional reservoir simulator: Part I-formulation and discretization. In: SPE Reservoir Simulation Symposium. Society of Petroleum Engineers (1997)
23. Watts, J.W., et al.: A compositional formulation of the pressure and saturation equations. SPE Reserv. Eng. **1**(03), 243–252 (1986)
24. Wheeler, M., Xue, G., Yotov, I.: A multipoint flux mixed finite element method on distorted quadrilaterals and hexahedra. Numer. Math. **121**(1), 165–204 (2011)
25. Wheeler, M.F., Xue, G., Yotov, I.: Accurate cell-centered discretizations for modeling multiphase flow in porous media on general hexahedral and simplicial grids. SPE J. **17**(03), 779–793 (2012)
26. Wheeler, M.F., Yotov, I.: A multipoint flux mixed finite element method. SIAM J. Numer. Anal. **44**(5), 2082–2106 (2006)
27. Zhang, D., Lu, Z.: An efficient, high-order perturbation approach for flow in random porous media via Karhunen–Loève and polynomial expansions. J. Comput. Phys. **194**(2), 773–794 (2004)