SPE 163599-PA

Fast Simulation of Polymer Injection in Heavy-Oil Reservoirs Based on Topological Sorting and Sequential Splitting

Knut-Andreas Lie, Halvor Møll Nilsen, Atgeirr Flø Rasmussen, Xavier Raynaud, SINTEF ICT.

This paper was prepared for presentation at the SPE Reservoir Simulation Symposium held in The Woodlands, Texas USA, 18-20 February 2013.

Abstract

We present a set of algorithms for sequential solution of flow and transport that can be used for efficient simulation of polymer injection modeled as a compressible two-phase system. Our formulation gives a set of nonlinear transport equations that can be discretized with standard implicit upwind methods to conserve mass and volume independent of the time step. In the absence of gravity and capillary forces, the resulting nonlinear system of discrete transport equations can be permuted to lower triangular form by using a simple topological-sorting method from graph theory. This gives an optimal nonlinear solver that computes the solution cell by cell with local iteration control. The single-cell systems can be reduced to a nested set of nonlinear scalar equations that can be bracketed and solved with standard gradient or root-bracketing methods. The resulting method gives orders-of-magnitude reduction in runtimes and increases the feasible time-step sizes. In fact, one can prove that the solver is unconditionally stable and will produce a solution for arbitrarily large time steps. For cases with gravity, the same method can be applied as part of a nonlinear Gauss–Seidel method. Altogether, our results demonstrate that sequential splitting combined with single-point upwind discretizations can become a viable alternative to streamline methods for speeding up simulation of advection-dominated systems.

1 Introduction

Heavy-oil resources are estimated to be more than twice the resources of conventional crude oil. Reservoirs containing heavy oil are challenging to produce, primarily because the oil has much higher viscosity than water, which will cause injected water to finger through the reservoir and leave a large percentage of the hydrocarbons behind as residual or non-recoverable oil (Gao 2011). Enhanced oil recovery is therefore essential to increase each field's lifetime and ultimately the world's oil producible resources.

Polymer flooding is a widely used EOR strategy (Xiaodong et al. 2011; Fletcher et al. 2012; Wassmuth et al. 2009), in which a dilute solution of a water-soluble polymer molecules is injected to increase the effective water viscosity and thereby establish more favorable mobility ratios between the injected and the displaced fluids. The addition of long-chain polymer molecules causes a reduction in the permeability and may also cause the water to behave like a non-Newtonian fluid, i.e., be resistant to flow at low velocities. This allows preferential filling of high-permeable zones and increases the areal sweep efficiency. Altogether, this results in a highly nonlinear fluid behavior and increased stiffness of the governing transport equations. In particular, because the water viscosity is strongly affected by the polymer concentration, it is crucial to capture polymer fronts sharply to resolve the nonlinear displacement mechanisms correctly.

Polymer fronts are, unlike water fronts, not self-sharpening and high numerical resolution is therefore required to limit the numerical diffusion that would otherwise bias or deteriorate simulation results. High numerical resolution can be achieved by using a fine grid, by introducing a higher-order discretization, or through a combination of these two approaches. Herein, we will focus on high grid resolution, and consider methods for simulating polymer flooding on high-resolution geo-cellular grid models, which are currently outside the reach of conventional solvers based on fully implicit discretizations. To enable simulation of large grid models, we apply a sequential solution method in which pressure and saturations/concentrations are updated in separate steps, e.g., as is commonly used in streamline simulators (Datta-Gupta and King 2007; Thiele et al. 2010). For the particular models considered herein, the splitting conserves mass and volume and has no theoretical restriction on the pressure and transport steps. This enables us to use efficient solvers that are specially targeted at the equations found in each of the sequential steps. For the pressure step, we use a standard two-point discretization, combined with highly efficient, algebraic multigrid, linear solvers. For the saturation/concentration equations, we apply an upstream mobility-weighted, finite-volume discretization in space and an implicit, first-order discretization in time. This choice may seem strange, given our goal of solving models with a high number

of cells. However, our motivation is that geo-cellular models tend to have large differences in face areas and cell volumes, which effectively preclude the use of explicit time stepping methods that are common for high-order discretization methods. Instead, we will develop a set of nonlinear solvers that are robust over a large span of CFL numbers and so fast that numerical diffusion can be reduced by using high grid resolution and time steps that are near the CFL limit of explicit schemes for the cells that contribute most to the numerical diffusion. The rationale is that first-order implicit and explicit schemes in practice will have equal numerical diffusion if the time step of the implicit method is approximately the same as the CFL restriction for explicit schemes in cells that neither have high flow nor small volumes. Moreover, localized methods like the discontinuous Galerkin method can be applied if higher approximation order is required for the spatial discretization, e.g., as discussed by Natvig and Lie (2008a).

The key to developing highly efficient transport solvers is the observation that the density contrast between the injected water and the oil is typically (much) smaller in the recovery of heavy oil than in traditional oil field operations, which implies a looser coupling between viscous and gravity forces. We exploit this loose coupling to introduce a splitting between advection and gravity segregation in the transport step. In the absence of gravity, the advective transport equations can be discretized by the first-order, implicit, upstream-mobility-weighted method to give a discrete nonlinear system. By performing a topological sort on the flux graph, the nonlinear system can be permuted to triangular form and solved cell-by-cell with local control over the nonlinear iterations (Natvig and Lie 2008a; Lie et al. 2012a). This reordering method is related to the potential ordering methods proposed by Kwok and Tchelepi (2007) and Shahvali and Tchelepi (2013), in which equations are reordered based on phase velocities rather than total velocity. When gravity is present, one can formulate an effective nonlinear Gauss–Seidel method that relies entirely on solving single-cell problems. Each single-cell problem can be solved using the Newton–Raphson method, or it can be reduced to a nested set of scalar nonlinear equations that can be bracketed and thus is guaranteed to have a unique solution for arbitrary large time steps. Altogether, this gives a very robust and efficient method that is fully competitive with state-of-the-art streamline methods (Thiele et al. 2010; Clemens et al. 2011; AlSofi and Blunt 2010), and exploits the same physical features of the governing equations. Our method, however, is based solely on standard finite-volume discretizations and hence avoids potential pitfalls like lack of mass conservation, allocation of fluxes to streamlines, mapping between streamlines and physical grid, and tracing of streamlines in irregular geometries, etc that complicate the implementation of streamline methods.

2 Mathematical Model and Discretization

In the following, we will consider a set of polymer models that have many of the features that are typically supported by contemporary commercial simulators. To this end, the polymer flooding process will be described by an immiscible, two-phase, three-component flow model that may include the effects of adsorption, rock and fluid compressibilities, pressure-dependent transmissibility, and gravity. Capillary effects are disregarded herein, but can be included by operator splitting in the same way as for streamline methods. In summary, the model is essentially the same as in a commercial simulator (Eclipse 2009). The numerical methods presented in the following can also apply to models that account for dead pore volume, i.e., that polymer cannot reach the smallest pores so that the effective pore volume $\phi(1 - S_{dpv})$ available for the flowing polymer solution is smaller than the pore volume ϕ of the rock. For brevity, this effect has been left out of the presentation.

2.1 Mathematical model.

To derive a model, we start from the conservation equations for oil, water, and polymer

$$\frac{\partial}{\partial t}(\rho_\alpha \phi S_\alpha) + \nabla \cdot (\rho_\alpha \vec{v}_\alpha) = 0, \qquad \alpha \in \{w, o\}, \tag{1a}$$

$$\frac{\partial}{\partial t}(\rho_w \phi S_w c) + \frac{\partial}{\partial t}\left(\rho_{r,\text{ref}}(1-\phi_{\text{ref}})\hat{a}\right) + \nabla \cdot (c\rho_w \vec{v}_{wp}) = 0. \tag{1b}$$

Here, ρ_α, $\vec{v}_\alpha$, and S_α denote density, velocity, and saturation of phase α; the porosity is denoted by ϕ and is assumed to be a function of pressure only; c denotes the polymer concentration; and $\vec{v}_{wp}$ the velocity of water containing diluted polymer. The function $\hat{a}$ models the amount of polymer that is adsorbed in the rock. The time scale of adsorption is much faster than that of mass transport and we will assume that adsorption takes place instantaneously so that $\hat{a}$ is a function of c only. The reference rock density is $\rho_{r,\text{ref}}$ and the reference porosity is ϕ_{ref}. Sources and sinks may be included in a manner equivalent to boundary conditions, and are left out of the above equations.

To get a solvable system, Eq. 1 must be supplied by a set of closure relations. Simple PVT behaviour is modeled through the formation-volume factors $b_\alpha = b_\alpha(p)$, defined by $\rho_\alpha = b_\alpha \rho_\alpha^S$, where ρ_α^S is the surface density of phase α. Inserting this into Eq. 1, the system can be simplified by dividing each equation with the relevant surface density ρ_α^S,

$$\frac{\partial}{\partial t}(b_\alpha \phi S_\alpha) + \nabla \cdot (b_\alpha \vec{v}_\alpha) = 0, \qquad \alpha \in \{w, o\}, \tag{2a}$$

$$\frac{\partial}{\partial t}(b_w \phi S_w c) + \frac{\partial}{\partial t}\left((1-\phi_{\text{ref}})a\right) + \nabla \cdot (b_w c \vec{v}_{wp}) = 0, \tag{2b}$$

where we for convenience have introduced the short-hand $a = \hat{a}\rho_{r,\text{ref}}/\rho_w^S$.

Darcy's law for the two phases can be written $\vec{v}_\alpha = -K\lambda_\alpha(\nabla p - b_\alpha(p)\rho_\alpha^S g\nabla z)$, where K denotes rock permeability, $\lambda_\alpha = k_\alpha/\mu_\alpha$ denotes the fluid mobility for phase α, g is the gravity constant, and z the coordinate in the vertical direction. We introduce the total mobility $\lambda = \lambda_o + \lambda_w$ and let $f_\alpha = \lambda_\alpha/\lambda$ denote the fractional flow of phase α. As long as the relative permeabilities k_α are nonnegative, monotone, and equal zero for $S_\alpha = 0$, the fractional flow functions $f_\alpha(S_\alpha)$ will be monotone and have end-point values $f_\alpha(0) = 0$ and $f_\alpha(1) = 1$. With this, the equation for the total velocity $\vec{v} = \vec{v}_w + \vec{v}_o$ reads

$$\vec{v} = -K\lambda\Big(\nabla p - \sum_\alpha b_\alpha \rho_\alpha^S f_\alpha g\nabla z\Big). \tag{3}$$

Next, we describe a simple fluid model for polymer diluted in water. The concentration of polymer is so small that the only physical property affected is the viscosity. To model the viscosity change of the mixture, we will use the Todd–Longstaff model (Todd and Longstaff 1972). This model introduces a mixing parameter $\omega \in [0,1]$ that takes into account the degree of mixing of polymer into water. Assuming that the viscosity μ_m of a fully mixed polymer solution is a function of the concentration, the effective polymer viscosity is defined as

$$\mu_{p,\text{eff}} = \mu_m(c)^\omega \mu_p^{1-\omega}, \qquad \mu_p = \mu_m(c_{\max}). \tag{4}$$

The viscosity of the partially mixed water is given in a similar way by

$$\mu_{w,e} = \mu_m(c)^\omega \mu_w^{1-\omega}. \tag{5}$$

The effective water viscosity $\mu_{w,\text{eff}}$ is defined by interpolating linearly between the inverse of the effective polymer viscosity and the partially mixed water viscosity

$$\frac{1}{\mu_{w,\text{eff}}} = \frac{1 - c/c_{\max}}{\mu_{w,e}} + \frac{c/c_{\max}}{\mu_{p,\text{eff}}}. \tag{6}$$

To take the incomplete mixing into account, we introduce the velocity of water that contains polymer, which we denote $\vec{v}_{wp}$. For this part of the water phase, the relative permeability is denoted by k_{rwp} and the viscosity is equal to $\mu_{p,\text{eff}}$. We also consider the total water velocity, which we still denote $\vec{v}_w$ and for which the viscosity is given by $\mu_{w,\text{eff}}$. Darcy's law then gives us

$$\vec{v}_w = -\frac{k_{rw}}{\mu_{w,\text{eff}} R_k(c)}\boldsymbol{K}(\nabla p_w - \rho_w g\nabla z), \qquad \vec{v}_{wp} = -\frac{k_{rwp}}{\mu_{p,\text{eff}} R_k(c)}\boldsymbol{K}(\nabla p_w - \rho_w g\nabla z) = m(c)\vec{v}_w, \tag{7}$$

as we assume that the presence of polymer does not affect the pressure and the density. We have also assumed that the relative permeability does not depend on mixing so that $k_{rwp} = k_{rw}$. The function $R_k(c)$ denotes the actual resistance factor and is a non-decreasing function that models the reduction of the permeability of the rock to the water phase due to the presence of absorbed polymer. Herein, we set $R_k(c) = 1 + (R_{rf} - 1)C_p^a(c)/C_p^a(c_{\max})$, where R_{rf} is the residual resistance factor defined as the ratio of the initial water mobility to the water solution mobility after polymer flooding, C_p^a is the amount of polymer absorbed, and $C_p^a(c_{\max})$ is the maximum possible absorbed polymer; all three depend on the rock type and are thus spatially dependent. A useful formula for $m(c)$ can easily be derived as follows:

$$m(c) = \frac{\mu_{w,\text{eff}}}{\mu_{p,\text{eff}}} = \Big[\Big(1 - \frac{c}{c_{\max}}\Big)\Big(\frac{\mu_p}{\mu_w}\Big)^{1-\omega} + \frac{c}{c_{\max}}\Big]^{-1}. \tag{8}$$

To simplify the notation in the following, we write Eq. 7 on the form,

$$\vec{v}_w = f(S,c)\vec{v} + \vec{g}_w, \qquad \vec{v}_{wp} = m(c)f(S,c)\vec{v} + \vec{g}_{wp}. \tag{9}$$

Here, the terms $\vec{g}_w$ and $\vec{g}_{wp}$ denote gravity contributions; their exact form will be given in Section 3.3 and is not important in the preceding discussion. **Fig. 1** show representative plots of the functions $f(S,c)$ and $m(c)c$ whose nonlinearity will play a prominent role in the following.

2.2 Discretization. To discretize Eq. 2, we first introduce a grid consisting of cells $\{C_i\}$ that each have a bulk volume V_i, integrate over each cell in space, and apply a backward-Euler discretization for the temporal derivative. This gives the discrete transport equations

$$\big(b_{\alpha,i}\phi_i S_{\alpha,i}\big)^{n+1} - \big(b_{\alpha,i}\phi_i S_{\alpha,i}\big)^n + \frac{\Delta t}{V_i}\sum_j \big(b_{\alpha,ij}v_{\alpha,ij}\big)^{n+1} = 0, \tag{10a}$$

$$\Big(b_{w,i}\phi_i S_{w,i}c_i + (1-\phi_{\text{ref},i})a_i\Big)^{n+1} - \Big(b_{w,i}\phi_i S_{w,i}c_i + (1-\phi_{\text{ref},i})a_i\Big)^n + \frac{\Delta t}{V_i}\sum_j \big(b_{\alpha,ij}c_{ij}v_{wp,ij}\big)^{n+1} = 0. \tag{10b}$$

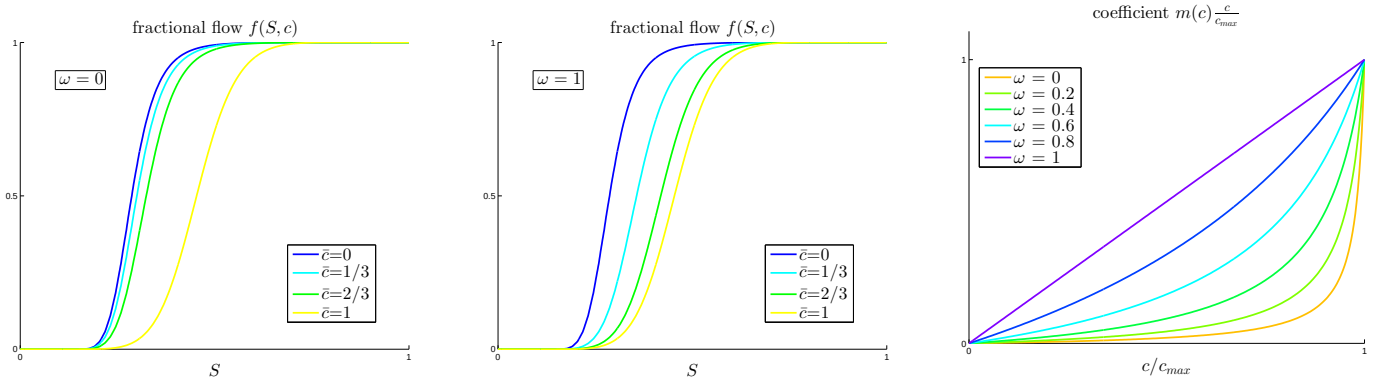

Fig. 1—Left: Plots of the fractional flow functions $f(S,c)$, for $\omega = 0$ and $\omega = 1$. Here, $\bar{c}$ denotes $\frac{c}{c_{max}}$. We observe, as expected, that the fractional flow is a decreasing function of the polymer concentration, which explains the use of polymer to increase oil recovery. When there is no polymer ($\bar{c} = 0$) or the solution is fully saturated with polymer ($\bar{c} = 1$), the fractional flow functions are independent of ω, but we notice that the decrease in fractional flow for intermediate values of the concentration is more important in the fully mixed case ($\omega = 1$); that is, the polymer is more effective in a fully mixed solution. Right: Plot of $m(c)c$. This coefficient is equal to the ratio between the polymer flux and the water flux. It can be seen as the analog of a relative permeability in a multiphase system, where concentration plays the role of saturation, see Booth (2008). From the plot, we observe that ω is directly related to the degree of nonlinearity contained in the model, with the linear case corresponding to a fully mixed solution ($\omega = 1$).

Compute initial conditions p_i^0, S_i^0, and c_i^0
foreach *time step* t^{n+1} **do**
  Compute pressure p_i^{n+1} and fluxes v_{ij}^{n+1} by solving Eq. 11 for all cells C_i
  Compute S_i^{n+1}, c_i^{n+1} by solving Eq. 10 (with $\alpha = w$ in Eq. 10a) for all cells C_i

Algorithm 1: Sequential splitting method for pressure and transport.

Here, subscripts i denote quantities associated with cell C_i and subscripts ij denote quantities associated with the interface between cells C_i and C_j. Superscripts denote time steps. To derive a discrete pressure equation, we sum the two phase equations, Eq. 10a, using the condition $S_w + S_o = 1$ to obtain

$$\phi_i^{n+1} - \phi_i^n \sum_\alpha \left(\frac{b_{\alpha,i}^n}{b_{\alpha,i}^{n+1}} S_{\alpha,i}^n \right) + \frac{\Delta t}{V_i} \sum_j \sum_\alpha \frac{b_{\alpha,ij}^{n+1}}{b_{\alpha,i}^{n+1}} \left(f_{\alpha,ij} v_{ij} + g_{\alpha,ij} \right)^{n+1} = 0. \tag{11}$$

Here, $f_{\alpha,ij}$ and $g_{\alpha,ij}$ denote upstream-weighted fractional-flow and gravity contributions of phase α, while v_{ij} denotes the flux corresponding to the total velocity $\vec{v}$, which can be expressed as $v_{ij} = T_{ij}(p_i - p_j + g_{ij})$, where T_{ij} denotes the transmissibility between cells C_i and C_j. The transmissibility is computed by the industry-standard approach in which the fluid mobility part is discretized with single-point, phase-based upstream weighting and the fluid-independent part by a standard two-point flux approximation that involves harmonic averaging. The equation necessary to compute discrete pressures and fluxes is now formed by combining Eq. 11 with the expression for the discrete flux.

3 Solution Strategy

Our overall system will consist of a pressure equation, Eq. 11, and two transport equations, Eq. 10a with $\alpha = w$ for the water saturation and Eq. 10b for the polymer concentration. To solve this coupled system, we use a sequential implicit solution procedure as outlined in **Algorithm** 1 that separates and solves the pressure and transport equations in consecutive steps. Herein, the pressure and transport equations are solved only once per time step, but if needed each time step can be iterated to reduce the (nonlinear) residuals to an arbitrarily small tolerance. In the pressure step, when solving Eq. 11, we use the value of saturation and concentration given at the previous time step so that the superscript $n + 1$ in the equation only refers to the pressure values.

The dynamics of the transport problem is generally determined by the balance between viscous and gravity forces. For heavy-oil systems, however, gravity segregation is typically a weak effect compared to viscous flow because of large injection rates and small differences in the densities of the oil and water phases. This means that the transport will mainly be co-current, giving a unidirectional flow property that can be exploited to develop efficient transport solvers. If gravity effects are negligible, one can prove that this sequential implicit splitting method will be unconditionally stable in the sense that it is mass conservative, preserves consistency between pressures and masses, has no time-step restriction between pressure and transport steps, and has no time-step restrictions on the transport steps. (The proof will be presented in a forthcoming paper.) In the next subsection, we

Reorder cells according to the fluxes $\{v_{ij}^{n+1}\}_{i=1}^{N}$
Compute S_i^{n+1}, c_i^{n+1} iteratively:
foreach *cell* C_i $i = \{1, \ldots, N\}$ **do**
  Solve the 2×2 single-cell problem defined by Eq. 12: find S and c such that $R_{w,i}^n(S, c) = 0$ and $R_{c,i}^n(S, c) = 0$.
  $(S, c) \rightarrow (S_i^{n+1}, c_i^{n+1})$

Algorithm 2: Advection step for a system that can be completely reordered.

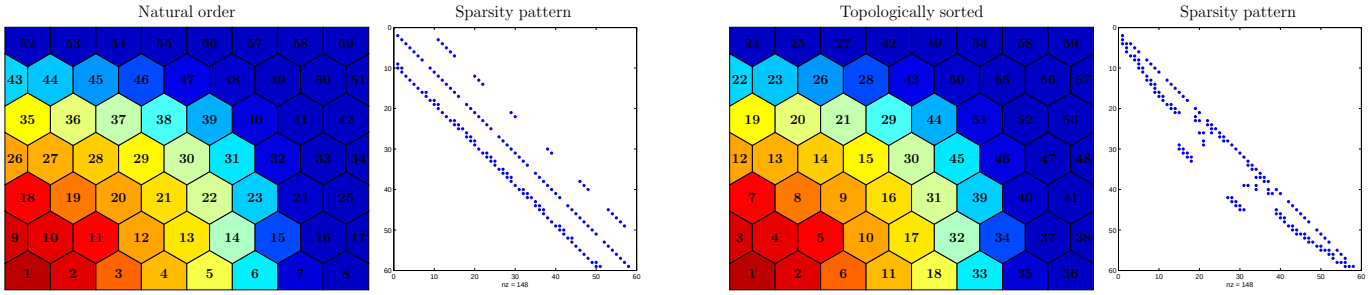

Fig. 2—Illustration of the reordering idea for a quarter-five spot simulation on a Voronoi grid with injection in the lower-left corner (cell 1) and production in the upper right (cell number 59). The two plots to the left show the natural numbering of the cells together with the resulting sparsity pattern of the discrete transport equations. The two plots to the right show the new ordering induced by the topological sort and the resulting sparsity pattern.

will also argue that each transport step can be solved using a finite number of operations, independent of the size of the time size of the splitting step $\Delta t = t^{n+1} - t^n$.

3.1 Advection step: the single-cell problem. If gravity is neglected and a monotone method like the standard two-point flux-approximation scheme is used to discretize the pressure equation, the velocity field $\vec{v}$ has no loops. The discrete equivalence of this property is that the directed graph describing the fluxes is acyclic and hence can be topologically sorted. Using the reordering induced by the topological sort, all cell neighbors of cell C_i can be divided into two index sets: $U(i)$ denotes all upwind neighbors and $D(i)$ denotes all downwind neighbors. If we use upstream-cell evaluation for the fractional flow function f_{ij}, the discrete transport equations can be written as two residual equations

$$\begin{aligned} R_{w,i}(S^{n+1}, c^{n+1}) =& \big(b_i\phi_i S_i\big)^{n+1} - \big(b_i\phi_i S_i\big)^{n} \\ &+ \frac{\Delta t}{V_i} \sum_{j\in U(i)} \big(f(S_j, c_j) b_{ij} v_{ij}\big)^{n+1} + \frac{\Delta t}{V_i} f(S_i, c_i)^{n+1} \sum_{j\in D(i)} \big(b_{ij} v_{ij}\big)^{n+1}, \end{aligned} \tag{12a}$$

$$\begin{aligned} R_{c,i}(S^{n+1}, c^{n+1}) =& \big(b_i\phi_i S_i c_i + a(c_i)(1-\phi_{\text{ref},i})\big)^{n+1} - \big(b_i\phi_i S_i c_i + a(c_i)(1-\phi_{\text{ref},i})\big)^{n} \\ &+ \frac{\Delta t}{V_i} \sum_{j\in U(i)} \big(m(c_j) c_j f(S_j, c_j) b_{ij} v_{ij}\big)^{n+1} + \big(m(c_i) c_i f(S_i, c_i)\big)^{n+1} \frac{\Delta t}{V_i} \sum_{j\in D(i)} \big(b_{ij} v_{ij}\big)^{n+1}. \end{aligned} \tag{12b}$$

All upwind cells C_j for $j \in U(i)$ will appear before cell C_i in the reordered numbering. If the residual equations in Eq. 12 are solved in the order given by a topologically sort of the total flux, the saturations S_j^{n+1} and concentrations c_j^{n+1} have already been computed when we visit cell C_i. Hence, the only unknowns in Eq. 12 are the values of saturation and concentration in the cell C_i, S_i^{n+1} and c_i^{n+1}. In other words, performing a topological sort enables us to permute the nonlinear system of discrete transport equations, Eq. 16, to a block-triangular form, as illustrated in **Fig. 2**. Because of the block-triangular form, the solution can be computed one cell at the time as outlined in **Algorithm 2**. This method was first introduced by Natvig et al. (2006, 2007) for linear transport equations and by Aarnes et al. (2006) for incompressible two-phase flow. The method was later extended to systems of transport equations and higher-order methods by Natvig and Lie (2008a). In particular, Natvig and Lie (2008b) demonstrated two orders-of-magnitude speedup for several examples of incompressible and slightly compressible two-phase flow. One order speedup was attributed to ordering of the equations, and one order was attributed to localization of the nonlinear iterations.

In an implicit method, all cells are solved simultaneously which means that the number of iterations is dictated by the cell that converges slowest. Algorithm 2 gives us local control over the iteration process, meaning that the number of iteration steps needed per cell is no longer forced to be the maximum needed by the worst cell. With local control, the average number of steps is much reduced, for example by exploiting that the residual will be identical to zero ahead of the displacement fronts in many flooding scenarios. Hence, no iterations are required in the corresponding cells. This is demonstrated in (Lie et al. 2012b). To

achieve further gains in efficiency compared with a straightforward Newton–Raphson method, we must devise an efficient method for solving each of the single-cell problems defined in Eq. 12. To simplify the presentation, we write the residual equations in a more compact form. That is, we let S and c denote S_i^{n+1} and c_i^{n+1} and introduce seven constants, $k_0, \ldots, k_6$, in which we collect terms involving cells C_j, $j \in U(i)$ that have already been computed, quantities that depend on the known pressure solution $(p_i^{n+1}, v_{ij}^{n+1})$ at time $n+1$, and terms from the previous time step n. Then, we can write Eq. 12 as a 2×2 nonlinear system of the form

$$R_w(S, c) = k_0 + k_1 S + k_2 f(S, c) = 0, \tag{13a}$$

$$R_c(S, c) = k_3 + k_4 Sc + k_5 a(c) + k_6 m(c) c f(S, c) = 0. \tag{13b}$$

Here, k_0 and k_3 are independent of Δt, while the remaining depend linearly, which means that the nonlinear parts of the system increase linearly in Δt.

The straightforward way of solving this system is by applying a gradient method like the Newton–Raphson method. Whereas such a method will be efficient in many cases, it is not guaranteed that the method will converge, or even that Eq. 13 has a unique solution. Fortunately, Lie et al. (2012b) recently proved that Eq. 13 has a unique solution for a simpler flow model without polymer adsorption and resistance factor. The key idea in the proof is to use Eq. 13a to formally eliminate S as a function of c, and then insert this function $S(c)$ into Eq. 13b to obtain a nonlinear scalar equation $R_c(S(c), c) = 0$. In general, $S(c)$ cannot be computed analytically, but must instead computed numerically by solving Eq. 13a for each fixed c. The resulting algorithm for the single-cell problem is hence a nested loop of iterations over nonlinear scalar equations $h(c) = 0$ that each can be solved robustly using bracketing scalar root-finding algorithms. If $h(a)$ differs in sign from $h(b)$, the function h crosses zero at least once in the interval $[a, b]$, which is thus guaranteed to contain a root. Given a valid initial interval, bracketing algorithms cannot fail to find a root if the function h is well behaved and will terminate in a finite number of iterations for a given tolerance on the residual. This means that the approximate solution can be computed in a finite number of iterations, which in practice will increase with the local CFL number. However, the nested bracketing algorithm guarantees that there is an upper bound that is *independent* of the size of the time step. The same can easily be proved for the system considered herein under the assumption of equal fluid compressibilities. The proof can also be extended to models with different fluid compressibilities provided care is taken when evaluating $b_{\alpha,ij}$ so that $b_{\alpha,ij} = b_\alpha(p_{ij}^{n+1})$ if $(f_o v_{ij} + g_{o,ij})$ and $p_{ij}^{n+1} - p_i^{n+1}$ have opposite sign, and $b_\alpha = b_\alpha(p_i^{n+1})$ otherwise. Likewise, the proof can be extended to models that account for dead pore volume (Eclipse 2009), provided that one makes certain modifications to eliminate the instability inherent in the model which allows for the computation of infinite polymer concentrations; details will be given in forthcoming papers.

There are several robust bracketing root-finding methods for solving scalar equations e.g., bisection or modified regula falsi, as well as hybrid methods like Brent's method that combine bisection, secant, and inverse quadratic interpolation. In our implementation we have used a modified regula falsi method described by Ford (1995), the Pegasus method, for both solves. This method only requires the evaluation of residuals, which are simple to implement and typically inexpensive to compute compared to the Jacobians required by gradient methods. Bracketing methods may not always have optimal convergence, but can still be used as nonlinear solvers in their own right or as robust fallback strategies for more efficient gradient-based methods when solving the single-cell problem, as demonstrated by Lie et al. (2012b).

3.2 Advection step: nonlinear Gauss–Seidel iterations. In the general case, we are not guaranteed that the total velocity $\vec{v}$ does not contain loops, i.e., blocks of mutually dependent cells that cannot be solved one by one in order. This happens in particular if gravity effects are included in the total velocity $\vec{v}$ or if a non-monotone method is used to discretize the pressure equation, Eq. 11. **Fig. 3** shows examples of loops occurring in discretized transport equations.

In (Natvig and Lie 2008b,a), the coupled multi-cell systems were solved using a straightforward stabilized Newton method. However, numerical experiments indicate that it is more efficient to use a simple iterated, nonlinear Gauss–Seidel in which we solve the multi-cell block one cell at a time, and repeat the process until an acceptable converged solution is obtained. This method will be outlined in the following. By applying the reordering, we ensure an optimal block structure of the nonlinear system in which the number of the loops and the size of each loop are minimized. The loop segments are solved in the correct order so that, for a given loop, the equations are well-posed and the only unknowns are the saturations and concentrations in the cells that belong to the loop. Let us now consider a loop of size N_ℓ. We drop the superscript $n+1$ and with a slight abuse of notation let $\boldsymbol{S} = [S_1, \ldots, S_{N_\ell}]$ and $\boldsymbol{c} = [c_1, \ldots, c_{N_\ell}]$ denote the values of saturation and concentration in the cells that belong to the loop. The residuals given by Eq. 12 take the following form

$$\boldsymbol{R}_w(\boldsymbol{S}, \boldsymbol{c}) = \boldsymbol{k}_0 + \boldsymbol{k}_1 \begin{bmatrix} S_1 \\ \vdots \\ S_{N_\ell} \end{bmatrix} + \boldsymbol{k}_2 \begin{bmatrix} f(S_1, c_1) \\ \vdots \\ f(S_{N_\ell}, c_{N_\ell}) \end{bmatrix}, \tag{14a}$$

$$\boldsymbol{R}_c(\boldsymbol{S}, \boldsymbol{c}) = \boldsymbol{k}_3 + \boldsymbol{k}_4 \begin{bmatrix} S_1 c_1 \\ \vdots \\ S_{N_\ell} c_{N_\ell} \end{bmatrix} + \boldsymbol{k}_5 \begin{bmatrix} a(c_1) \\ \vdots \\ a(c_{N_\ell}) \end{bmatrix} + \boldsymbol{k}_6 \begin{bmatrix} m(c_1) c_1 f(S_1, c_1) \\ \vdots \\ m(c_{N_\ell}) c_{N_\ell} f(S_{N_\ell}, c_{N_\ell}) \end{bmatrix} \tag{14b}$$

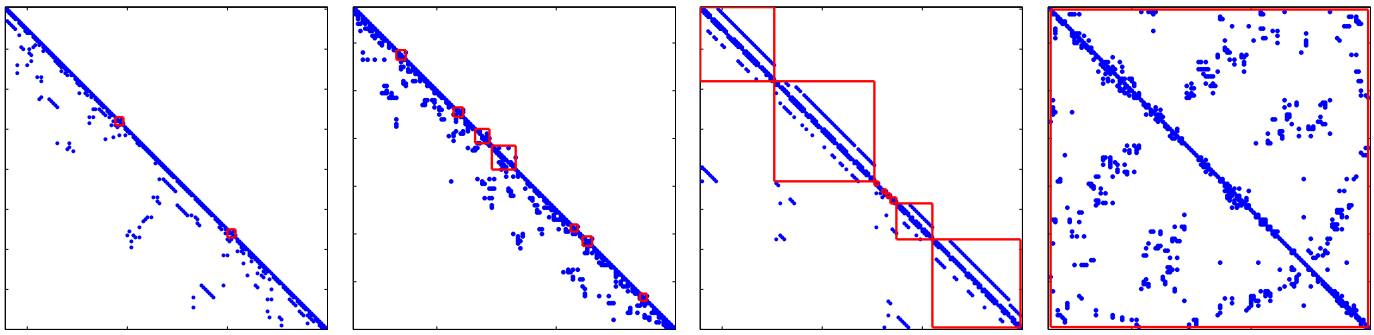

Fig. 3—Illustration of loops occuring in discretized transport equations. The plots show the sparsity pattern of the nonlinear system for four different cases with increasing influence of loops, from left to right: (i) an almost completely reorderable system containing only two small loops, (ii) a minor fraction the cells involved in seven small loops, (iii) the majority of cells involved in eight loops, and (iv) all cells connected in one big loop. When solving the nonlinear system, the local residual equations for the cells inside the red squares are coupled and must be solved simultaneously.

where $\boldsymbol{k}_0$, $\boldsymbol{k}_3$ are vectors and $\boldsymbol{k}_1$, $\boldsymbol{k}_2$, $\boldsymbol{k}_4$, $\boldsymbol{k}_5$, $\boldsymbol{k}_6$ are matrices that do not depend on $\boldsymbol{S}$ and $\boldsymbol{c}$ and are defined analogously as in Eq. 13. In the nonlinear Gauss–Seidel method, we loop through all the cells that belong to the loop and in each cell C_i solve the local 2×2 nonlinear single-cell problem to update S_i and c_i, using values from previous time or iteration step in all upwind cells. This process is repeated until the residuals are below a prescribed tolerance. Let superscript k denote the iterations of the Gauss–Seidel algorithm. Then, for $k = 1, 2, \ldots$, we compute sequentially S_i^k and c_i^k for $i = 1, \ldots, N_\ell$ by solving the single-cell problems

$$0 = R_{w,i}^k(S,c) = R_{w,i}(S_1^k, \ldots, S_{i-1}^k,\ S,\ S_{i+1}^{k-1}, \ldots, S_{N_\ell}^{k-1},\ c_1^k, \ldots, c_{i-1}^k,\ c,\ c_{i+1}^{k-1}, \ldots, c_{N_\ell}^{k-1}), \tag{15a}$$

$$0 = R_{c,i}^k(S,c) = R_{c,i}(S_1^k, \ldots, S_{i-1}^k,\ S,\ S_{i+1}^{k-1}, \ldots, S_{N_\ell}^{k-1},\ c_1^k, \ldots, c_{i-1}^k,\ c,\ c_{i+1}^{k-1}, \ldots, c_{N_\ell}^{k-1}). \tag{15b}$$

The single-cell problem of Eq. 15 takes the form of Eq. 13 and can be solved by a Newton–Raphson method or in the same robust manner as described in the previous subsection. The convergence of the method can be established for small time steps, but the proof is omitted for brevity. The proof also shows that the convergence speed increases when the dependence of the residuals with respect to the components belonging to the cells downward in the loop is small, that is,

$$\left|\frac{\partial R_{w,i}}{\partial S_j}\right| + \left|\frac{\partial R_{w,i}}{\partial c_j}\right| \ll 1 \quad \text{and} \quad \left|\frac{\partial R_{c,i}}{\partial S_j}\right| + \left|\frac{\partial R_{c,i}}{\partial c_j}\right| \ll 1 \quad \text{for all } i \text{ and } j > i.$$

In the special case in which the cells $C_1, \ldots, C_{N_\ell}$ are reorderable, we have $\frac{\partial R_{w,i}}{\partial S_j} = \frac{\partial R_{w,i}}{\partial c_j} = \frac{\partial R_{c,i}}{\partial S_j} = \frac{\partial R_{c,i}}{\partial c_j} = 0$ for all i and $j > i$ and the solution can then be found in exactly one iteration. The necessary steps for solving the advection problem are summarized in **Algorithm** 3.

Reorder cells according to the fluxes $\{v_{ij}^{n+1}\}_{i=1}^N$
→ A sequence of blocks B_n of mutually dependent cells, for $n = 1, \ldots, N_B$
for $\ell = 1, \ldots, N_B$ **do**
  if *block* B_ℓ *contains only one cell* **then**
    Solve single-cell problem Eq. 13 for the cell in B_ℓ.
  else
    Apply the non linear Gauss–Seidel algorithm to the block B_ℓ:
    while *cell residuals exceed tolerance* **do**
      for $n = 1, \ldots, N_\ell$ **do**
        Solve single-cell problem Eq. 15 for cell C_n in block B_ℓ

Algorithm 3: Advection step for a system that can only be partially reordered.

3.3 Gravity splitting. If gravity effects cannot be neglected, we need to introduce an additional operator splitting for the transport equations, Eq. 10, to be able to utilize the reordering methods discussed in the two previous subsections. This operator splitting method was first introduced within streamline simulation (Gmelig Meyling 1990, 1991; Bratvedt et al. 1996), but can also offer certain benefits for finite-volume methods, e.g., as discussed in (Lie et al. 2012a).

First, we solve the advective equations that account for viscous effects (for brevity subscripts w have been dropped)

$$\big(b\phi S^*\big)_i^{n+1} - \big(b\phi S\big)_i^n + \frac{\Delta t}{V_i}\sum_j \big(bf(S^*,c^*)v\big)_{ij}^{n+1} = 0, \tag{16a}$$

$$\Big(b\phi c^* S^* + (1-\phi_{\text{ref}})a(c^*)\Big)_i^{n+1} - \Big(b\phi cS + (1-\phi_{\text{ref}})a(c)\Big)_i^n + \frac{\Delta t}{V_i}\sum_j \big(bm(c^*)c^* f(S^*,c^*)v\big)_{ij}^{n+1} = 0, \tag{16b}$$

using the Gauss–Seidel method described in Algorithm 3. This gives us intermediate saturation and concentration distributions $S^{*,n+1}$ and $c^{*,n+1}$, which are then used as initial conditions for a second set of segregation equations that account for gravity effects

$$\big(b\phi(S-S^*)\big)_i^{n+1} + \frac{\Delta t}{V_i}\sum_j \big(b\lambda_o f(S,c)(\rho_w-\rho_o)gK\nabla z\big)_{ij}^{n+1} = 0, \tag{17a}$$

$$\Big(b\phi c(S-S^*) + (1-\phi_{\text{ref}})(a(c)-a(c^*))\Big)_i^{n+1} + \frac{\Delta t}{V_i}\sum_j \big(bm(c)c\lambda_o f(S,c)(\rho_w-\rho_o)gK\nabla z\big)_{ij}^{n+1} = 0. \tag{17b}$$

In the segregation equation, all cells will in principle be coupled and must be solved for simultaneously using a Newton–Raphson method. However, stratigraphic models are often built so that they have skewed pillars near faults and vertical pillars elsewhere. For models that only contain vertical cell columns, the segregation equations will decouple between columns and can be solved one column at the time. Linearizing the segregation equation inside a single column leads to a tridiagonal system that can be solved very by Gaussian elimination. Likewise, numerical experiments show that using a nonlinear Gauss–Seidel algorithm, similar to the one outlined in Section 3.2, can also be very efficient for cases with small density differences. To this end, we visit the cells inside the column in an alternating pattern, sweeping both from above and below. Solving for multiple columns is an operation that is straightforward to parallelize. In the general case, sets of cells that can be solved independently (e.g., in parallel) can be identified as connected components in the cell graph defined such that cells are connected across a face if $\vec{n}\cdot\vec{g} \neq 0$. Then, one can use a similar nonlinear Gauss–Seidel method to iterate over all cells inside each connected component.

Finally, we note that by summing Eq. 16 and Eq. 17 we obtain

$$\big(b\phi S\big)_i^{n+1} - \big(b\phi S\big)_i^n + \frac{\Delta t}{V_i}\sum_j \Big(\big(bf(S^*,c^*)v\big)_{ij}^{n+1} + \big(b\lambda_o f(S,c)(\rho_w-\rho_o)gK\nabla z\big)_{ij}^{n+1}\Big) = 0, \tag{18a}$$

$$\begin{aligned}\Big(b\phi cS + (1-\phi_{\text{ref}})a(c)\Big)_i^{n+1} - \Big(b\phi cS + (1-\phi_{\text{ref}})a(c)\Big)_i^n + \\ \frac{\Delta t}{V_i}\sum_j \Big(\big(bm(c^*)c^* f(S^*,c^*)v\big)_{ij}^{n+1} + \big(bm(c)c\lambda_o f(S,c)(\rho_w-\rho_o)gK\nabla z\big)_{ij}^{n+1}\Big) = 0,\end{aligned} \tag{18b}$$

which implies, after summing over all the cells, that

$$\sum_i \big(b\phi S\big)_i^{n+1} = \sum_i \big(b\phi S\big)_i^n, \qquad \text{and} \qquad \sum_i \big(b\phi cS + (1-\phi_{\text{ref}})a(c)\big)_i^{n+1} = \sum_i \big(b\phi cS + (1-\phi_{\text{ref}})a(c)\big)_i^n. \tag{19}$$

In other words, the total masses of water and polymer are conserved. The total mass of oil is in general not conserved and this discrepancy can be seen as a splitting error of the pressure and transport equations. In the special case of equal compressibilities, that is $b_o = b_w$, the total mass conservation of oil is recovered, as it follows from the discrete pressure equation Eq. 11 and the total mass conservation of water Eq. 19.

4 Numerical examples

In this section will present a few numerical experiments that demonstrate the utility of our numerical strategy and verify and validate the corresponding simulator that was implemented as part of the Open Porous Media (OPM) initiative (OPM 2012). The simulator is available as free and open-source code and can be downloaded from the OPM website under the GPLv3 license. For simplicity, all simulations reported in the following were performed on a single core. In the following examples, all 2×2 single-cell problems are either solved by the bracketed solver discussed above or by a Newton–Raphson solver with the nested bracketing method as a fallback if Newton iterations fail.

A standard Newton–Raphson method for the full polymer system would consist of linearizing Eq. 10, solving the resulting $2N \times 2N$ system, and iterating. We do not make a direct comparison of the standard Newton–Raphson and the reordered solvers herein, but instead refer the reader to (Natvig and Lie 2008b) for a comparison for the two-phase case without polymer. Here, one order of magnitude of speedup was observed by using reordering for the *linearized* Newton–Raphson equations, and an additional

order by using reordering to localize the nonlinear iterations. Since the polymer equations represent a more complex nonlinear system, one can expect at least similar speedup.

The overall computational method involves several types of time steps that must be specified for the sequential splittings and implicit discretizations: one for the sequential splitting of pressure and transport, one for the sequential splitting of the transport step into advection and segregation steps, one for the implicit advection solver, one for the segregation solver, and one for the compressible pressure solver. These time steps need not be equal, but must have a certain internal consistency to ensure that the outer time steps of the sequential splittings are multiples of the inner time steps of the sub-equations. Moreover, to get an optimal algorithm, each step size should be chosen to reflect the time scales (or stiffness/nonlinearity) of the corresponding coupling or subequation. In practice, one will always choose the time step of the pressure/transport splitting and the pressure solver to be the same; for guidelines of how to choose this time step, we refer the reader to the literature on streamlines methods. For each pressure step, one may perform one or more transport steps that each may consist of one or more advection and segregation steps. This is discussed in more detailed in (Lie et al. 2012a). Herein, all time steps are assumed to be equal for simplicity, unless explicitly stated otherwise.

4.1 Example 1: Polymer slug in a 1D reservoir. In the first example, we will verify our simulator against a commercial simulator (Eclipse 2009) for three different polymer models of increasing complexity. To this end, we consider a 1D horizontal and homogeneous, 1500 m long reservoir that is initially filled with an oil with viscosity 5 cP. To produce the oil, we first inject water with viscosity 0.5 cP from the left side for 300 days, and then a polymer slug for 500 days, before continuing injection of pure water for another 2000 days. The polymer will increase the water viscosity by a factor twenty. The base-case model uses linear relative permeabilities as reported in **Table 1**, water viscosity 0.5 cP, oil viscosity 5.0 cP, no adsorption, no dead pore space, and a Todd–Longstaff mixing parameter $\omega = 1$. In the second model, we include adsorption as given in Table 1 and dead pore space equal to 0.15. In the third model, the mixing parameter is set to $\omega = 0.5$. The computed solutions plotted in **Fig. 4** show excellent agreement for all three cases, with only a slight deviation for the third model. For the last case, we also include solutions for a series of runs with longer time steps, which shows that one can increase the time step so that the polymer front travels 1–2 cells per time step without adversely affecting the resolution of the polymer slug. Notice that this corresponds to a CFL number that is significantly larger than the explicit limit for the whole system.

TABLE 1—Fluid data for Example 1.

Relative permeabilities		
S_w	k_{rw}	k_{ro}
0.2	0.0	1.0
0.7	0.7	0.0
1.0	1.0	0.0

Viscosity	
c	$\mu_m(c)/\mu_w$
0.0	1.0
7.0	20.0

Adsorption	
c	$\hat{a}(c)$
0.0	0.0000
2.0	0.0015
8.0	0.0025

Compressibility	
Phase	Value (bar^{-1})
rock	$3.00 \cdot 10^{-6}$
water	$3.03 \cdot 10^{-6}$
oil	$1.25 \cdot 10^{-7}$

4.2 Example 2: Diagonal flow in a 3D cube. In the second example, we will investigate how our numerical methods scale with an increasing number of grid cells. To this end, we consider an idealized, synthetic test case consisting of a homogeneous reservoir in the form of a cube described on a grid with $n \times n \times n$ grid cells. The purpose of the example is to investigate how the transport solver scales with the number of cells, and hence we disregard gravity and use a constant number of ten time steps for all grid resolutions. (In a real physical case, the extent to which gravity can be neglected would, of course, depend upon the strength of the injection rate relative to the gravity forces.) A total of one pore volume of water is injected in the cell at the bottom south-west corner and fluids are produced from the cell at the upper north-east corner. Polymer is injected in the period between 0.2 and 0.5 PVI. No-flow boundary conditions are prescribed on all outer boundaries. The fluids are described using the base-case model from the previous example.

Fig. 5 reports the CPU times for the pressure and transport solvers. The pressure solver uses the algebraic multigrid linear solver from `dune:istl` (Blatt and Bastian 2007). As expected, the runtime of the pressure solver does not scale linearly with the number of cells, but increases with an average factor of 1.13. (This factor would most likely be higher for highly heterogeneous petrophysical data). The transport solver, on the other hand, scales linearly because the fluxes do not contain any loops and hence the discrete transport equations can be permuted to a sequence of 2×2 single-cell problems.

4.3 Example 3: Gravity convection. We have tested the performance of the nonlinear Gauss-Seidel solver for the transport step for a worst-case scenario: a gravity convection cell in which all grid cells are mutually dependent. We have a 100m $\times$ 100m domain in the xz-plane, with constant permeability equal to 0.1 Darcy, unit porosity, and two phases with densities 1000 kg/m^3 and 600 kg/m^3. Initially, the light and heavy fluids occupy the left and right parts of the domain, gravity will then seek to redistribute the fluids so that the heavy fluid ends up at the bottom and the light one on top, alike to a rotational motion. There are no sources or sinks, and no polymer. We have run all cases both with the Gauss-Seidel-based transport solver (including segregation by operator splitting), and a traditional Newton-Raphson solver.

In **Fig. 6** the left plot shows the effect of increasing spatial resolution. The case is run for 2000 days in 10 uniform time steps.There is an increase in computational cost per cell as we increase resolution. The primary reason for this is the increased

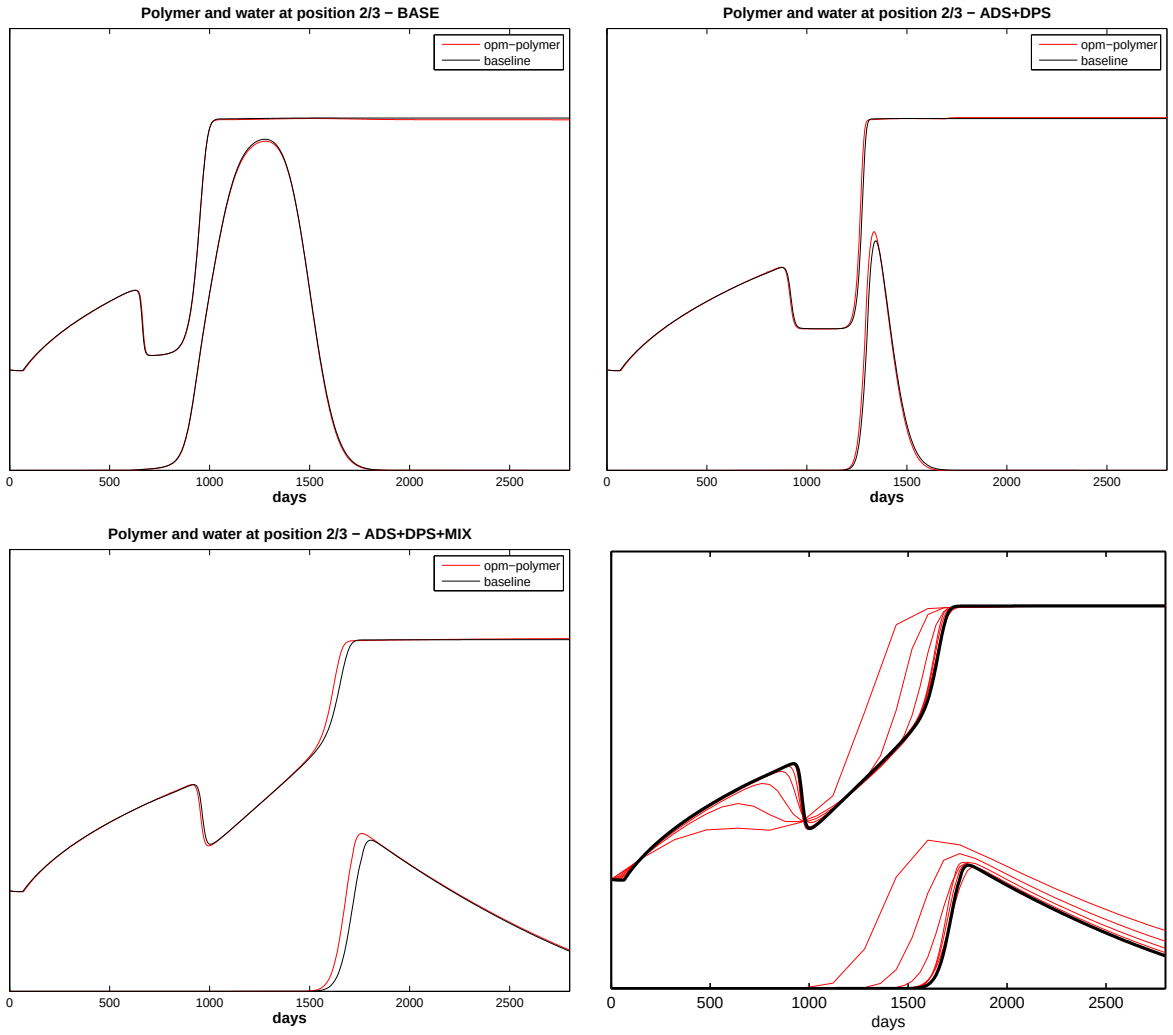

Fig. 4—Comparison of 1D polymer slug computed by the OPM polymer solver (red lines) and a commercial simulator (black lines) for three different model configurations: base-case model, base-case model with adsorption and dead pore space, and with mixing parameter $\omega = 0.5$. All simulations used five-day time steps. The lower-right plot also shows solutions computed with time steps of 10, 20, 40, 80, and 160 days for the OPM solver.

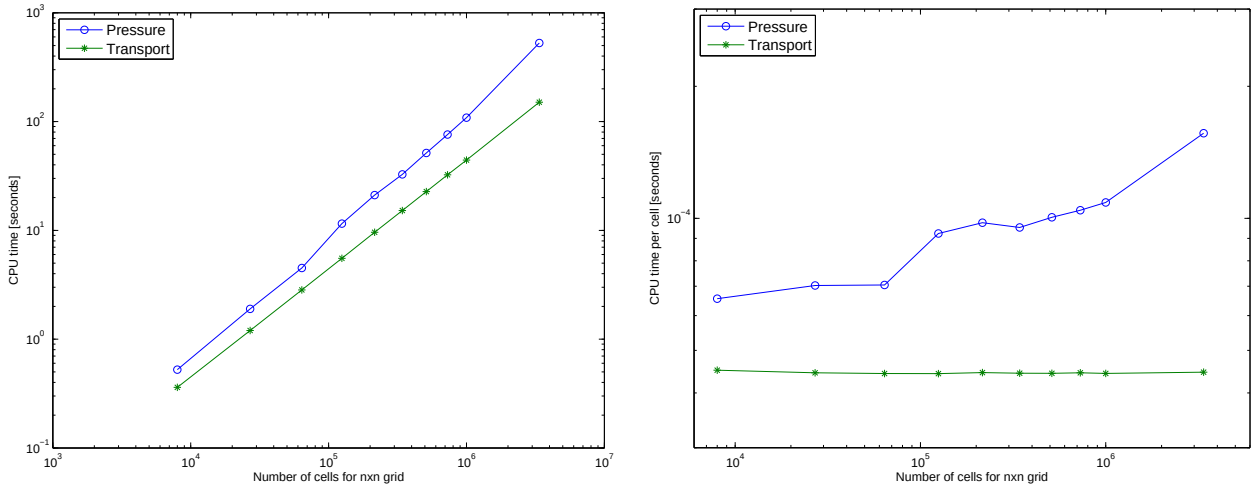

Fig. 5—CPU times for the $n \times n \times n$ reservoir in Example 2. The left plot shows total CPU time as a function of the size of the model. The right plot shows the CPU time per cell as a function of the size of the model.

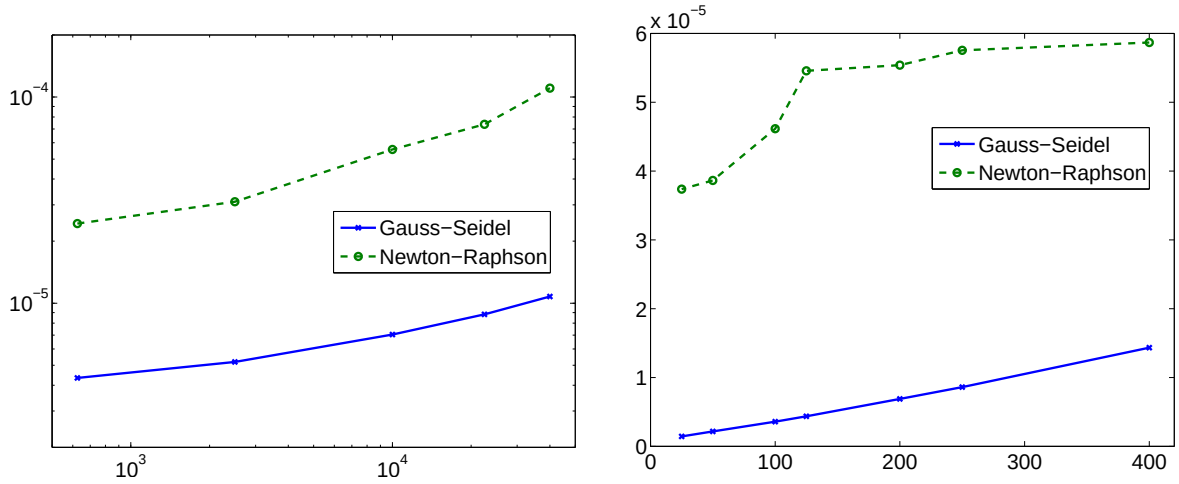

Fig. 6—CPU times per cell per timestep used by the transport solvers in Example 3. Left: logarithmic plot of runtime in seconds versus the number of cells. Right: logarithmic plot of runtime in seconds versus the length of the time step.

number of iterations needed, both by the Gauss-Seidel and Newton algorithms, as the number of cells in the loop increase. The average number of iterations for Gauss-Seidel are reported in the upper table of **Table 2**. One may note that the increase in cost is not as large as the increase in number of iterations. This is because a sparse updating procedure is used for Gauss-Seidel, at each iteration marking cells that have upstream saturation changes above a certain threshold (10^{-9} was used for this case). Only the marked cells are then updated at the next iteration, which rapidly reduces the cost of successive iterations.

TABLE 2—Average number of iterations required by Gauss-Seidel solver in Example 3.

Size	25×25	50×50	100×100	200×200
Average iterations	15.5	28.7	65	111

Time steps	5	8	10	16	20	40	80
Average iterations	190	134	111	80	76	69	25

The right-hand plot of **Fig. 6** shows the effect of increasing the number of time steps. The case is run for 2000 days with a varying number of uniform time steps on a 100×100 cell grid. As expected, reducing the time step reduces the computational cost per cell significantly. This effect is so large that it may even take less time in total to use more time steps. With shorter time steps, a smaller part of the total domain undergoes significant changes. Fewer iterations are then needed for convergence, as seen in the lower table of **Table 2**.

For both comparisons, the Gauss-Seidel method works well: it is stable and quite fast, more so than the Newton method, which starts to suffer convergence failures for the largest timesteps. The Newton method implemented is very straightforward, in particular no attempt has been made to order the linearized equations, but that would probaly not have made a difference in this particular test case, which is designed to be non-reorderable.

4.4 Example 4: Polymer front resolution. The aim of our work is to make an efficient solver that can be used to reduce computational costs, or alternatively for the same computational time increase the spatial resolution to decrease the error from numerical diffusion. In this example, we highlight the latter with an idealized example consisting of a 1000 m, horizontal, 1D reservoir with uniform permeability 0.1 Darcy and uniform porosity of 0.5, except in the interval between 50 and 100 where the porosity is 0.005. The purpose of this region is to mimic the effect of small grid cells or fast flow near wells, which will always appear in real reservoir simulations. The fluids have a viscosity contrast of 10 and polymer modifies the water viscosity by a factor 5 at maximum concentration.

Fig. 7 reports simulations with time steps corresponding to CFL numbers $\nu \approx 10$, 1, and 0.1 in the high-porosity region. As expected, our implicit method is accurate for $\nu \leq 1$ in the high porosity region, despite that this corresponds to one hundred times larger CFL numbers and hence very large time steps in the low-porosity region. The reason why the solver works so well is that the main contribution to the numerical diffusion will come from the high-porosity region where the fronts travel longest in time. To see this, we can look at the worst-case scenario of a passively advected wave, $\phi u_t + a u_x = 0$, which would correspond to the advection of the polymer concentration for $\omega = 1$. First, we transform the equation to time-of-flight coordinates (t, τ), where $\tau = x\,\phi$. Then, we observe that the effective equation for our numerical discretization is given as $\partial_t u + a\partial_\tau u = \frac{1}{2}(a^2\Delta t + a\Delta\tau)\partial_\tau^2 u$. This means that a discontinuity propagated by the implicit advection scheme for a time t

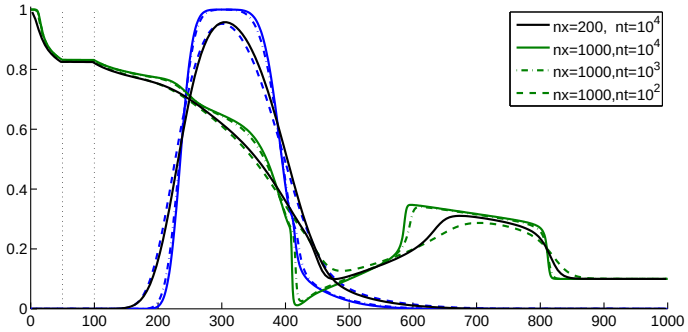

Fig. 7—Comparison of 1D polymer slug calculation using different number of time steps. The colored lines represent water saturation (green) and polymer concentration (blue) computed using 10^2 time steps (dashed), 10^3 time steps (dash-dotted) and 10^4 time steps on a grid with 1000 cells. The black lines represent a simulation with 10^4 time steps and 200 cells. The total time is five years. The case has uniform porosity of 0.5, except in the interval [50,100], where the porosity is 0.005.

will be smeared to a width $\mathcal{O}(\sqrt{a(a\Delta t + \Delta\tau)\cdot t})$. In our case, we have two different regions: a region of length ℓ_L with low porosity ϕ_L, and a region of length ℓ_H with high porosity ϕ_H. Since the time step and the grid sizes are constant for the whole domain, we have that $(a\Delta t + \Delta x\phi_L) < (a\Delta t + \Delta x\phi_H)$. The residence time in each of the regions are $t_L = \tau_L/a = \ell_L\phi_L/a$ and $t_H = \tau_H/a = \ell_H\phi_H/a$. This means that the smearing will be dominated by the region having largest accumulated time-of-flight. For a front that travels from $x = 0$ to $x = 500$, we have that $\tau_H = 450 \cdot 0.5 = 900 \cdot \tau_L = 50 \cdot 0.005$. In other words, the main contribution to the numerical smearing will come from the high-porosity region in which the CFL number, and hence the numerical smearing, is low. This argument can easily be extended to more realistic setups in three dimensions and suggests that an optimal solution strategy for an implicit discretization of the advection step is obtained by minimizing the CFL number and the grid size, measured in terms of time-of-flight, in the regions that contribute most to the accumulated time-of-flight for a given traveling wave.

For explicit methods, the time step is severely restricted by fast flow in small cells or fast flowing regions, like around wells, even if these regions contribute little to the solution in terms of time-of-flight. Regridding is often not an option since the small grid cells are introduced to represent important flow connections in the system and the fast flowing regions need to keep their resolution to minimize the error in the pressure solver. Compared with an explicit scheme, our implicit method can achieve less numerical diffusion by using the performance gained through the use of long time steps in the fast-flowing regions to increase the resolution in slow-flowing regions that dominate the numerical diffusion of wave propagating through the reservoir. This stipulated gain in accuracy is, of course, case specific and will depend on the ratio between the largest CFL numbers and the CFL number in the regions corresponding to the largest accumulation in time-of-flight. Compared to traditional implicit methods, our method can use the orders-of-magnitude efficiency gain to reduce time step and cell size in the same regions. The need for higher grid resolution is clearly seen in by comparing the solid lines in Fig. 7 (black lines: 200 cells, blue/green lines: 1000 cells), which both use very small time steps. Another interesting feature is seen by comparing the dashed and the black lines. For the black line (200 cells), the numerical diffusion is dominated by the grid cells, while for the dashed lines (1000 cells), the error is dominated by the time step and is more sensitive to front speeds.

4.5 Example 5: The Norne field. To validate our method, we consider a reservoir model representing Norne, a Norwegian Sea reservoir. The experiments presented in the following represent a plausible simulation case, but do *not* represent a real polymer injection operation. The Norne model (IO Center, NTNU 2012) is only used to provide realistic reservoir geometry, petrophysical parameters, and well positions. Fluid properties and well schedules are synthetic, but based on realistic heavy-oil cases from elsewhere in the world. We emphasize that as far as the authors know, Statoil and the Norne license partners have no plans to inject polymer into the field as discussed herein.

The Norne model shown in **Fig. 8** consists of approximately 45000 cells and is a faulted corner-point grid with fairly complex and heterogeneous geometry: the ratio of cell volumes between the largest and smallest cell is $3.6 \cdot 10^5$, while the ratio between largest and smallest interface area is $2.7 \cdot 10^9$. Most cells have six neighbours, but some have as many as 21, due to faults. The permeability ranges from 0.3 mD to 4 darcies. The model contains twenty wells, of which six are injectors and fourteen are producers. The reservoir is initially filled with oil and connate water, and simulated allowing for compressible rock and fluids. For the fluid model, we use the data described in **Table 3** and Fig. 8, water and oil densities of 962 and 1080 kg/m^3, water and oil viscosities of 0.48 and 180 cP, and a Todd–Longstaff mixing parameter $\omega = 1$. In this model, the pillars of the corner-point grid are skewed, and the segregation equations are therefore globally coupled. We have made the approximation when solving the segregation of treating the columns as uncoupled from each other. We believe this is reasonable because of small density differences, but the accuracy of the assumption is not investigated in detail herein.

We compare a plain water injection scenario lasting 4000 days to a polymer injection scenario in which a polymer solution

TABLE 3—Fluid data for Example 5.

Viscosity	
c	$\mu_m(c)/\mu_w$
0	1.0
0.5	3.6
1.0	6.3
1.5	12.5
2.0	25.8
3.0	48.0

Adsorption			
c	$\hat{a}(c)$	c	$\hat{a}(c)$
0.000	0.000000	1.250	0.000019
0.250	0.000010	1.500	0.000019
0.500	0.000012	1.750	0.000020
0.750	0.000016	2.000	0.000030
1.000	0.000018	3.000	0.000030

Compressibility	
Phase	Value (bar^{-1})
rock	$3.0 \cdot 10^{-6}$
water	$5.0 \cdot 10^{-5}$
oil	$5.0 \cdot 10^{-5}$

dead pore space: 0.05
residual resistivity factor: 1.3

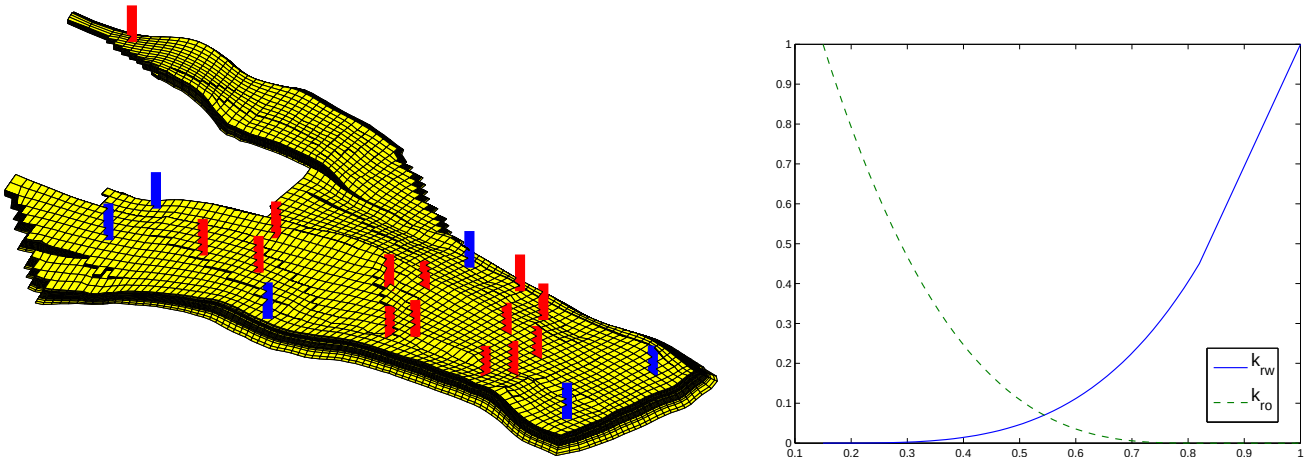

Fig. 8—Left plot: the Norne model with injection wells shown in blue and production wells shown in red. Right plot: relative permeability curves.

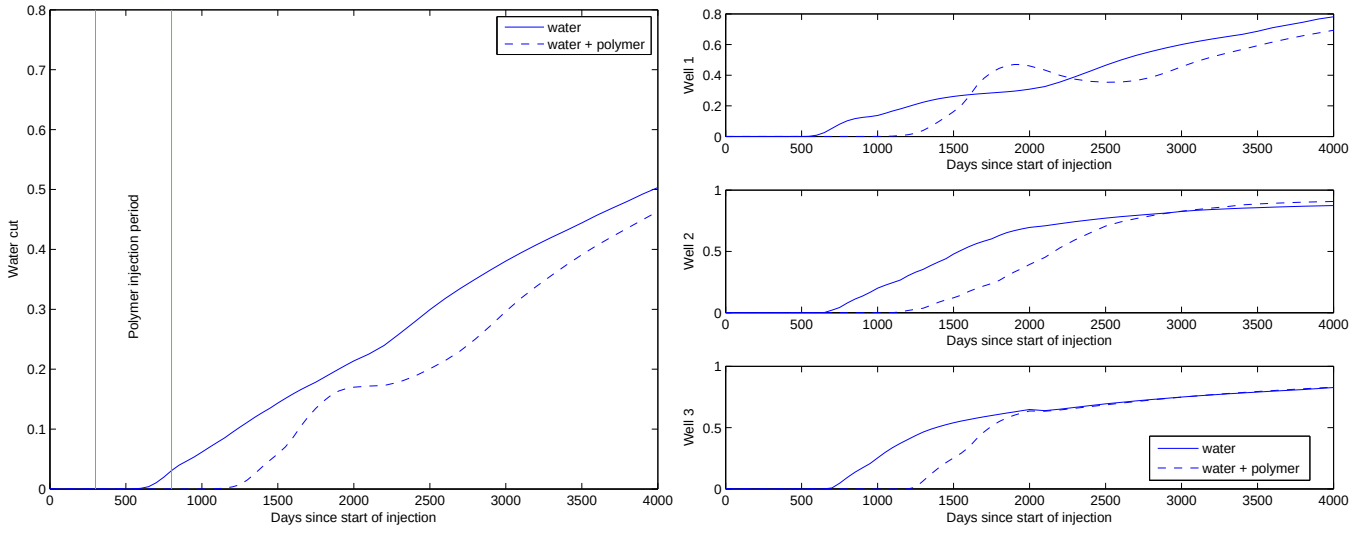

Fig. 9—Comparison of water-cut curves for the Norne test case with water injection and polymer flooding.

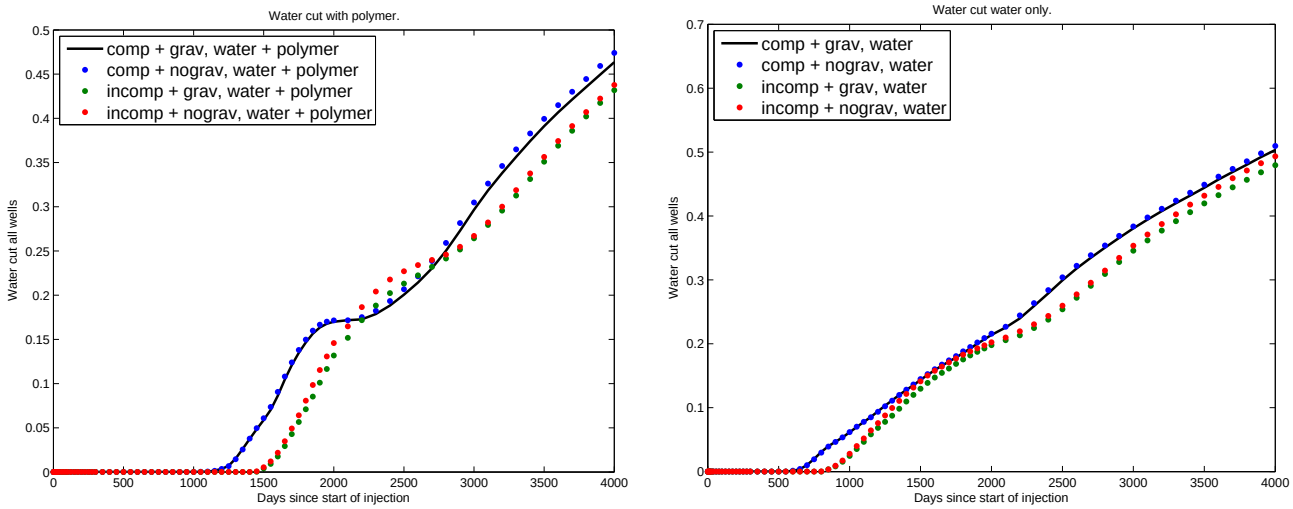

Fig. 10—Comparison of water-cut curves for the base-case model and models without gravity and/or compressibility for polymer flooding (left) and water injection (right).

TABLE 4—CPU times in seconds for the Norne model measured using a single core on 2.5 GHz Intel i7 processor.

	Compressible model		Incompressible model	
Case	Polymer flooding	Water injection	Polymer flooding	Water injection
Pressure solver	100 sec	99 sec	27 sec	27 sec
Transport solver	19 sec	12 sec	19 sec	11 sec
– without gravity	7.8 sec	5.7 sec	3.8 sec	2.4 sec

is injected between days 300 and 800. The 4000 days have been computed over 73 steps with time-steps ranging from 1 day initially, to 100 days in the latter half. The time steps were chosen small initially to establish the displacement fronts in the near-well areas. As the fronts move into the reservoir, the time step is increased. Beyond this, the steps were chosen quite arbitrarily. In the left part of **Fig. 9** we have plotted the total water cut of both scenarios, while the right part shows water-cut curves for the three producers with earliest breakthrough. The plots clearly demonstrate that the polymer has the effect of delaying water breakthrough and changing the shape of the curves.

The CPU time of the two simulations are reported in **Table 4**, where we clearly see that the computation time is dominated by the pressure part which takes five times longer than the transport solver for polymer flooding and eight times longer for water flooding. For the transport solver, a large part of the runtime is used by the gravity segregation solver; running with no gravity reduces the runtime by a factor 2–3 in both cases. The effect of gravity is small in this case because the density difference between the two phases is only 12%, and **Fig. 10** shows that the water-cut curves with and without gravity are almost indistinguishable. Neglecting gravity should therefore be a reasonable assumption.

The second effect that contributes to increase the runtime is compressibility. With incompressible rock and fluids, the pressure equation becomes linear and does not require any iterations. As a result, the runtime is reduced by a factor 3.7 compared with the nonlinear pressure solves in the compressible case. The water-cut curves are similar to the compressible case, but the water breakthrough comes significantly later. Allowing for rock compressibility with the porosity multiplier restricted to being a linear function of pressure gives curves closer to the base case, while keeping the low computational cost of a linear pressure equation.

4.6 Example 6: A heavy-oil field. As a last validation test, we consider a realistic case based on a real-field model of a heavy-oil reservoir. The model consists of 600 000 cells and has approximately forty wells. Petrophysical data, fluid properties, and well positions are from the real model, but the injection strategy and well controls are purely synthetic. To produce oil, polymer is injected along with water from all injection wells at a concentration that increases the water viscosity by a factor fifty. Polymer and water are assumed to be perfectly mixed (i.e., we use $\omega = 1$).

Fig. 11 shows oil saturation, polymer concentration (> 0.1) and the number of iterations used (> 0) in the last time step, with the outer surfaces of the reservoir superimposed. To compute the solution, we use 10 time steps of 100 days, and the bracketing solver for the single-cell problems. Fig. 11 reports the corresponding number of iterations used in the last time step for all methods. A key observation from the figure is that since the injection fronts only contact a relatively small number of cells, the nonlinear iterations will be effectively localized to regions around wells and near the oil-water contact, where either the water saturation or the polymer concentration changes. This should lead to significant reduction in computational cost compared with a global Newton–Raphson solve.

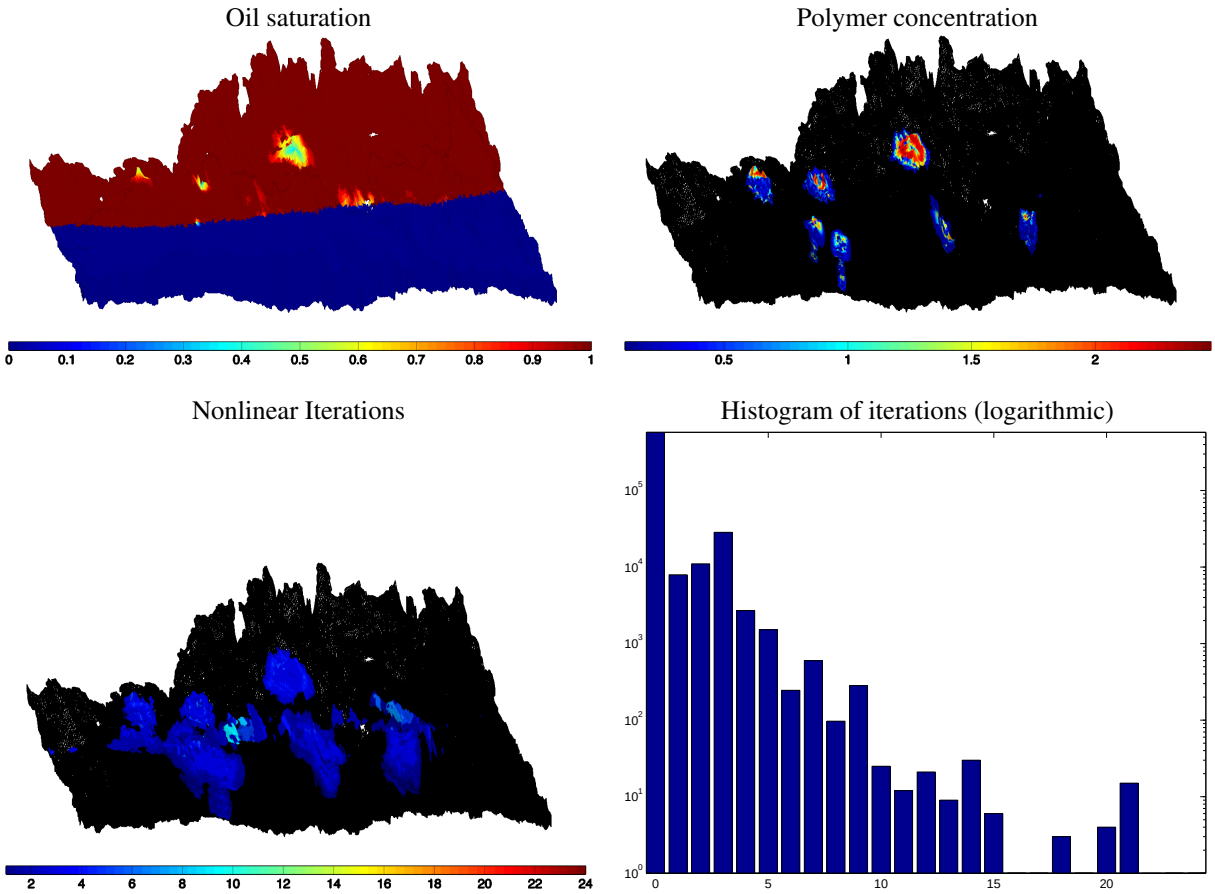

Fig. 11—Simulation of polymer injection for a real-field model. All plots show quantities from the last step in a simulation of 1000 days.

For the pressure step, we use AGMG (AGMG 2012; Notay 2010) as linear solver since it turned out to be slightly more efficient than `dune:istl` for this particular case. Each pressure solve converges within four nonlinear iterations and takes approximately 10 seconds on a Linux workstation with a 2.6 GHz Intel i7 processor. Motivated by the previous example (and preliminary numerical experiments), we neglect the influence of gravity. With this assumption, one transport step takes approximately 0.5 seconds. This means that we obtain twenty transport steps for the cost of a single pressure step. Since the coupling of the pressure and transport is not very tight, it can be advantageous to compute multiple time-steps in the advective solver for each pressure step to increase the resolution of polymer fronts without significantly affecting the total runtime. These results are encouraging, but further research is needed to find an optimal time-stepping strategy for achiving the best accuracy for the given computational cost.

5 Concluding remarks

We have presented a highly efficient simulation method for realistic models of polymer flooding in heavy-oil reservoirs.The new simulation method is implemented as part of the Open Porous Media (OPM) initiative and is freely available from the website (OPM 2012) under the GPLv3 license. The current software implements a polymer model that includes dead pore space, adsorption, and permeability reduction. The simulator is particularly efficient for systems in which rock compressibility dominates fluid compressibilities and gravity only governs slow processes; such systems are highly relevant for enhanced recovery of heavy oil.

The key points to the efficiency of our new method are: (i) to exploit the loose coupling between flow and transport to enable special solvers for the respective equations; (ii) to exploit the weak influence of gravity and split the transport calculation into an advective and a segregation part; and (iii) to localize the nonlinear solves of the advective part of transport step to significantly reduce computational costs. Initial testing indicates that the method works well for a wide range of time steps and is robust with respect to large differences in pore volumes and face areas that are typically present in realistic high-resolution geo-cellular models. In particular, if nested bracketing is used for solving the localized single-cell problems, either as a full solver or as a fallback strategy, we have an unconditionally stable method that avoids time-step restrictions induced by stability problems. However, further research is needed to understand what the feasible time-step ranges are for practical reservoir simulation in

terms of accuracy versus efficiency.

Research is also needed to develop efficient localization strategies for the nonlinear iterations for flow fields that contain large loops that may for instance be caused by gravity effects. As we have demonstrated above, neglecting gravity effects will in certain cases give a significant speedup without having an adverse effect on the overall accuracy of the simulation. However, we are not aware of any criterion that can accurately and robustly determine whether gravity effects can be neglected or not. Currently, this would typically be decided by considering density ratios or by simulating spatial or temporal subsets of the model. We refer the interested reader to a recent paper (Lie et al. 2012a) for a preliminary discussion of how gravity affects the efficiency of the type of operator-splitting methods presented herein.

The reordering method discussed herein can be very fast when run on a single core. However, since the implicit updating procedure needs to visit cells sequentially in a predetermined order, the resulting methods are predominantly serial and not necessarily straightforward to implement efficiently on a parallel computer. The most obvious approach to this end would be to exploit the delineation of the reservoir volume that is inherent in the reordering procedure. The reordering method can easily be extended to label cells with the injection wells (or inflow boundaries) they are influenced by (see e.g., (Shahvali et al. 2012)), and by using these labels, one can identify independent regions that can be computed concurrently. This way, each concurrent thread can start in a unique well and march the solution outwards until one encounters cells that are also influenced by one of the other threads. While this obviously would not provide perfect scaling across a large number of cores, it could potentially provide sufficient speedup on workstation computers with tens of cores, provide the model contains a sufficient number of injection wells.

6 Acknowledgments

The authors would like to thank Statoil Petroleum A/S for funding and access to data sets for simulator testing. The authors would also like to thank Statoil (operator of the Norne field) and its license partners ENI and Petoro for the release of the Norne data. Further, the authors acknowledge the Center for Integrated Operations at NTNU for cooperation and coordination of the Norne Cases. Finally the authors thank Ove Sævareid for valuable input and Eclipse simulations, and Vegard Kippe, Alf Birger Rustad, Stein Krogstad, Bård Skaflestad, and Kristin M. Flornes for fruitful discussions and feedback on our work.

References

Aarnes, J. E., Krogstad, S., Lie, K.-A., and Natvig, J. R. 2006. Fast sequential implicit porous media flow simulations using multiscale finite elements and reordering of cells for solution of nonlinear transport equations. In *Proceedings of ECMOR X (10th European Conference on the Mathematics of Oil Recovery)*, Amsterdam, The Netherlands. ECMOR.

AGMG 2012. Iterative solution with AGgregation-based algebraic MultiGrid. `http://homepages.ulb.ac.be/~ynotay/AGMG/`.

AlSofi, A. M. and Blunt, M. J. 2010. Streamline-based simulation of non-Newtonian polymer flooding. *SPE J.*, 15(4):895–905.

Blatt, M. and Bastian, P. 2007. The iterative solver template library. In Kåström, B., Elmroth, E., Dongarra, J., and Wasniewski, J., editors, *Applied Parallel Computing. State of the Art in Scientific Computing*, volume 4699 of *Lecture Notes in Scientific Computing*, pages 666–675. Springer Verlag.

Booth, R. 2008. *Miscible Flow Through Porous Media*. University of Oxford.

Bratvedt, F., Gimse, T., and Tegnander, C. 1996. Streamline computations for porous media flow including gravity. *Transp. Porous Media*, 25(1):63–78.

Clemens, T., Abdev, J., and Thiele, M. R. 2011. Improved polymer-flood management using streamlines. *SPE J.*, 14(2):171–181.

Datta-Gupta, A. and King, M. J. 2007. *Streamline Simulation: Theory and Practice*, volume 11 of *SPE Textbook Series*. Society of Petroleum Engineers.

Eclipse 2009. *Eclipse Technical Description Manual*. Schlumberger, 2009.2 edition.

Fletcher, P., Cobos, S., Jaska, C., , Forsyth, J., , Crabtree, M., and Canada, N. G. 2012. Improving heavy oil recovery using an enhanced polymer system. In *SPE Improved Oil Recovery Symposium, 14-18 April 2012, Tulsa, Oklahoma, USA*.

Ford, J. A. 1995. Improved algorithms of illinois-type for the numerical solution of nonlinear equations. Technical Report CSM-257, University of Essex.

Gao, C. H. 2011. Scientific research and field applications of polymer flooding in heavy oil recovery. *J. Petrol. Explor. Prod. Technol.*, 1:65–70.

Gmelig Meyling, R. H. J. 1990. A characteristic finite element method for solving non-linear convection-diffusion equations on locally refined grids. In Guerillot, D. and Guillon, O., editors, *2nd European Conference on the Mathematics of Oil Recovery*, pages 255–262, Arles, France. Editions Technip.

Gmelig Meyling, R. H. J. 1991. Numerical methods for solving the nonlinear hyperbolic equations of porous media flow. In *Third International Conference on Hyperbolic Problems, Vol. I, II (Uppsala, 1990)*, pages 503–517, Lund. Studentlitteratur.

IO Center, NTNU 2012. The Norne benchmark case. url: `http://www.ipt.ntnu.no/~norne/wiki/doku.php`.

Kwok, F. and Tchelepi, H. 2007. Potential-based reduced Newton algorithm for nonlinear multiphase flow in porous media. *J. Comput. Phys.*, 227(1):706–727.

Lie, K.-A., Natvig, J. R., and Nilsen, H. M. 2012a. Discussion of dynamics and operator splitting techniques for two-phase flow with gravity. *Int. J Numer. Anal. Mod. (Special issue in memory of Magne Espedal)*, 9(3):684–700.

Lie, K.-A., Nilsen, H. M., Rasmussen, A. F., and Raynaud, X. 2012b. An unconditionally stable splitting method using reordering for simulating polymer injection. In *ECMOR XIII – 13th European Conference on the Mathematics of Oil Recovery, Biarritz, France, 10-13 September 2012*, number B25.

Natvig, J. R. and Lie, K.-A. 2008a. Fast computation of multiphase flow in porous media by implicit discontinuous Galerkin schemes with optimal ordering of elements. *J. Comput. Phys.*, 227(24):10108–10124.

Natvig, J. R. and Lie, K.-A. 2008b. On efficient implicit upwind schemes. In *Proceedings of ECMOR XI, Bergen, Norway, 8–11 September*. EAGE.

Natvig, J. R., Lie, K.-A., and Eikemo, B. 2006. Fast solvers for flow in porous media based on discontinuous Galerkin methods and optimal reordering. In Binning, P., Engesgaard, P., Dahle, H., Pinder, G., and Gray, W., editors, *Proceedings of the XVI International Conference on Computational Methods in Water Resources*, Copenhagen, Denmark.

Natvig, J. R., Lie, K.-A., Eikemo, B., and Berre, I. 2007. An efficient discontinuous Galerkin method for advective transport in porous media. *Adv. Water Resour.*, 30(12):2424–2438.

Notay, Y. 2010. An aggregation-based algebraic multigrid method. *Electron. Trans. Numer. Anal.*, 37:123–140.

OPM 2012. The Open Porous Media (OPM) initiative. url: `http://www.opm-project.org/`.

Shahvali, M., Mallison, B., Wei, K., and Gross, H. 2012. An alternative to streamlines for flow diagnostics on structured and unstructured grids. *SPE J.*, 17(3):768–778.

Shahvali, M. and Tchelepi, H. 2013. Efficient coupling for nonlinear multiphase flow with strong gravity. In *SPE Reservoir Simulation Symposium, The Woodlands, TX, USA, 18–21 February 2013*. SPE 163659-MS.

Thiele, M. R., Batycky, R. P., Pöllitzer, S., and Clemens, T. 2010. Polymer-flood modeling using streamlines. *SPE J.*, 13(2):313–322.

Todd, M. R. and Longstaff, W. J. 1972. The development, testing, and application of a numerical simulator for predicting miscible flood performance. *J. Petrol. Tech.*, 24(7):874–882.

Wassmuth, F. R., Green, K., Arnold, W., and Cameron, N. 2009. Polymer flood application to improve heavy oil recovery at East Bodo. *J. Can. Petrol. Technol.*, 48(2):55–61.

Xiaodong, K., Jian, Z., Fujie, S., Fengjiu, Z., Guozhi, F., Junru, Y., Xiansong, Z., and Wentao, X. 2011. A review of polymer EOR on offshore heavy oil field in Bohai Bay, China. In *SPE Enhanced Oil Recovery Conference, 19-21 July 2011, Kuala Lumpur, Malaysia*.

Nomenclature

Subscripts:
α = phase ($\alpha = w$ for water, $\alpha = o$ for oil)
i = cell number
i, j = face number (interface between cell i and j)

Superscript:
n = time step

Physical quantities:
p = pressure
p_α = pressure of the phase α
S_α = Saturation of the phase α
c = polymer concentration
$\vec{v}_\alpha$ = flux of the phase α
$\vec{v}_{wp}$ = polymer flux
ρ_α = density of the phase α
ρ_α^S = density of the phase α at surface condition
$\rho_{r,\text{ref}}$ = reference rock density
ϕ = porosity
ϕ_{ref} = reference porosity
$c_{\max}$ = maximum polymer concentration
$\hat{a}$ = adsorption polymer concentration
a = normalized adsorption polymer concentration
b_α = formation volume factor of the phase α
K = permeability tensor
$k_{r\alpha}$ = relative permeability of the phase α
k_{rwp} = relative permeability of polymer
μ_m = viscosity of the fully mixed polymer solution
μ_α = viscosity of the phase α
$\mu_{p,\text{eff}}$ = effective polymer viscosity
$\mu_{w,e}$ = viscosity of the partially mixed water
$\mu_{w,\text{eff}}$ = effective water viscosity
ω = mixing parameter
R_k = actual resistance factor
R_{rf} = residual resistance factor
R_α = residual value for the conservation equation of the phase α
R_c = residual value for the conservation equation of the polymer
Δt = time step
V_i = volume of the cell i