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

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 two-phase system with rock compressibility and equal fluid compressibilities. 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 splitting is unconditionally stable and 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. 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 standard 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 [9]. 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 [7, 23, 24], 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 the 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 [6, 21] or the chemical simulator UTCHEM. For the particular models considered herein, the splitting conserves mass and volume and has no 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 a highly efficient, algebraic multigrid, linear solver. For the saturation/concentration equations, we apply a standard, upstream mobility-weighted, finite-volume discretization in space and 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 cell volumes and face areas that 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 with high flow and large volumes. Moreover, localized methods like the discontinuous Galerkin method can be applied if higher approximation order is required for the spatial discretization.

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 a standard, implicit, upstream mobility-weighted method to give a discrete nonlinear system that can be permuted to triangular form and solved cell-by-cell with local control over the nonlinear iterations [13, 14, 16]. 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 a standard 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 [2, 5, 21], and exploits the same physical features of the governing equations. Our method, however, is based solely on standard finite-volume discretizations and hence avoids certain difficulties that are inherent in streamline methods: lack of mass conservation, allocation of fluxes to streamlines, mapping between streamlines and physical grid, tracing of streamlines in irregular geometries, etc.

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 compressibility, identical phase compressibilities, pressure-dependent transmissibility, gravity, but capillary effects are not included. The assumption of equal fluid compressibilities is essential to devise an unconditionally stable, sequential splitting method, but except for that, the model is essentially the same as in a commercial simulator [20]. The results presented in the following can also be extended to models that account for dead pore volume, but this effect has been left out for brevity.

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 $\phi(p)$ of pressure only, c 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 higher 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 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. We make the simplifying assumption that $b_o = b_w = b$, so that the phases have the same compressibility behaviour. Inserting this into Eq. 1, the system can be simplified by dividing each equation with the relevant surface density ρ_α^S,

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

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

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 $\vec{v}_\alpha = -K\lambda_\alpha(\nabla p - b(p)\rho_\alpha^S g \nabla z)$, 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 λ_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

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

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 [22]. 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

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

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. 10 for all cells C_i
  Compute S_i^{n+1}, c_i^{n+1} by solving Eq. 9 (with $\alpha = w$ in Eq. 9a) for all cells C_i

Algorithm 1: Sequential splitting method for pressure and transport.

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. A useful formula for $m(c)$ can easily be derived as follows:

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

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 standard implicit method for the temporal derivative. This gives the discrete transport equations

$$\big(b_i\phi_i S_{\alpha,i}\big)^{n+1} - \big(b_i\phi_i S_{\alpha,i}\big)^{n} + \frac{\Delta t}{V_i}\sum_j \big(b_{ij} v_{\alpha,ij}\big)^{n+1} = 0, \tag{9a}$$

$$\Big(b_i\phi_i S_{w,i} c_i + (1-\phi_{\text{ref},i})a_i\Big)^{n+1} - \Big(b_i\phi_i S_{w,i} c_i + (1-\phi_{\text{ref},i})a_i\Big)^{n} + \frac{\Delta t}{V_i}\sum_j \big(b_{ij} c_{ij} v_{wp,ij}\big)^{n+1} = 0. \tag{9b}$$

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. 9, using the condition $S_w + S_o = 1$ to obtain

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

where v_{ij} denotes the flux corresponding to the total velocity $\vec{v}$. This flux can be expressed as $v_{ij} = T_{ij}(p_i - p_j + g_{ij})$ if we discretize Eq. 3 by a standard two-point flux approximation. Here, T_{ij} denotes the transmissibility between cells C_i and C_j. The equation necessary to compute discrete pressures and fluxes is now formed by combining Eq. 10 with the expression for the discrete flux.

3 Solution Strategy

Our overall system will consist of a pressure equation, Eq. 10, and two transport equations, Eq. 9a with $\alpha = w$ for the water saturation and Eq. 9b for the polymer concentration. To solve this coupled system, we use a standard sequential solution procedure as outlined in **Algorithm** 1 that separates and solves the pressure and transport equations in consecutive steps.

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

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. 11: 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..

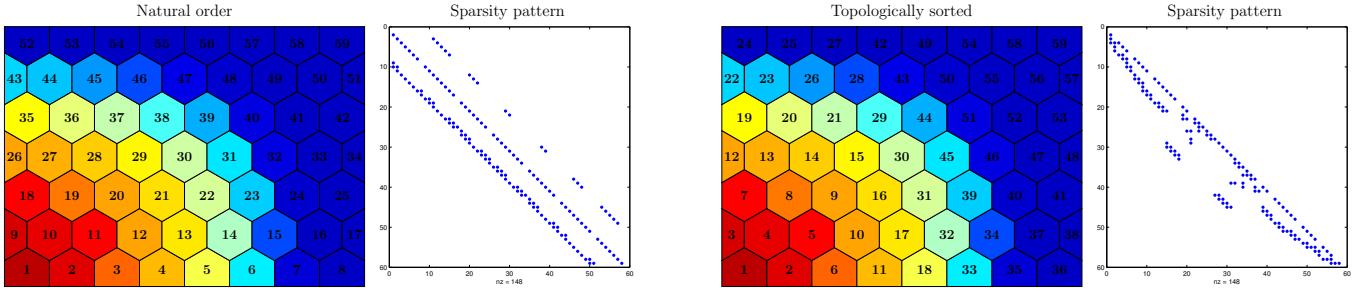

Fig. 1—Illustration of the reordering idea for a quarter-five spot simulation on a Voronoi grid. The two plots to the left show the natural numbering of the cells together with the resulting sparsity pattern of the discrete transport equations. The two plots to the right show the new ordering induced by the topological sort and the resulting sparsity pattern.

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, this sequential 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. In the next subsection, we will also show 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 neighbours of cell C_i can be divided into two index sets: $U(i)$ denotes all upwind neighbors of cell C_i 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

$$R_{w,i}(S^{n+1}, c^{n+1}) = \big(b_i\phi_i S_i\big)^{n+1} - \big(b_i\phi_i S_i\big)^{n} + \frac{\Delta t}{V_i}\sum_{j\in U(i)}\big(f(S_j, c_j)b_{ij}v_{ij}\big)^{n+1} + \frac{\Delta t}{V_i}f(S_i, c_i)^{n+1}\sum_{j\in D(i)}\big(b_{ij}v_{ij}\big)^{n+1}, \tag{11a}$$

$$R_{c,i}(S^{n+1}, c^{n+1}) = \big(b_i\phi_i S_i c_i + a(c_i)(1-\phi_{\text{ref},i})\big)^{n+1} - \big(b_i\phi_i S_i c_i + a(c_i)(1-\phi_{\text{ref},i})\big)^{n} + \frac{\Delta t}{V_i}\sum_{j\in U(i)}\big(m(c_j)c_j f(S_j, c_j)b_{ij}v_{ij}\big)^{n+1} + \big(m(c_i)c_i f(S_i, c_i)\big)^{n+1}\frac{\Delta t}{V_i}\sum_{j\in D(i)}\big(b_{ij}v_{ij}\big)^{n+1}. \tag{11b}$$

All upwind cells $C_j, j \in U(i)$ will appear before cell C_i in the reordered numbering, and if the residual equations in Eq. 11 are solved in the topologically sorted order given by the flux, the saturation and concentration values S_j^{n+1} and c_j^{n+1} have already been computed when we visit cell C_i. Hence, the only unknowns in Eq. 11 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. 15, to a block-triangular form, as illustrated in **Fig. 1**. 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 and Lie [16, 17], who demonstrated two orders-of-magnitude speedup for several examples of incompressible two-phase flow.

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, as for a standard implicit method solving for all cells simultaneously. 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 [15]. To achieve further gains in efficiency compared with a standard Newton–Raphson method, we

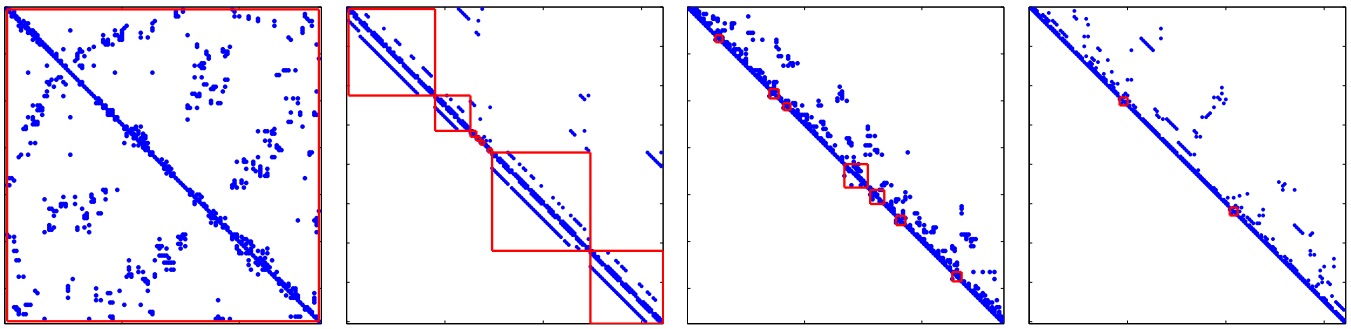

Fig. 2—Illustration of loops occuring in discretized transport equations. The plots show the sparsity pattern of the nonlinear system for four different cases with diminishing influence of loops, from left to right: (i) all cells connected in one big loop, (ii) the majority of cells involved in eight loops, (iii) a minor fraction the cells involved in seven small loops, and (iv) an almost completely reorderable system containing only two small loops. When solving the nonlinear system, the local residual equations for the cells inside the red squares are coupled and must be solved simultaneously.

must devise an efficient method for solving each of the single-cell problems defined in Eq. 11. We start by simplify the residual equations. 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 that have already been computed (i.e., C_j for $j \in U(i)$), 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. 11 as a 2×2 nonlinear system of the form

$$R_w(S, c) = k_0 + k_1 S + k_2 f(S, c) = 0, \tag{12a}$$

$$R_c(S, c) = k_3 + k_4 Sc + k_5 a(c) + k_6 m(c) c f(S, c) = 0. \tag{12b}$$

The straightforward way of solving this system is by applying a gradient method. However, it is not obvious that this method will converge, or even that Eq. 12 has a unique solution. Fortunately, Lie et al. [15] recently proved that Eq. 12 has a unique solution for a simpler flow model without polymer adsorption and resistance factor, and that this solution can be computed in a finite number of iterations, *independently* of the size of the time step. The key idea in the proof is to use Eq. 12a to formally eliminate S as a function of c, and then insert this function $S(c)$ into Eq. 12b 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. 12a 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. The same result can be proved for the system considered herein and the proof can also be extended to models that account for dead pore volume [20], 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 [8], 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. [15].

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. 10. **Fig. 2** shows examples of loops occurring in discretized transport equations.

In [16, 17], the coupled multi-cell systems were solved using a standard 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. 11 take the following form

$$\boldsymbol{R}_w(\boldsymbol{S}, \boldsymbol{c}) = \boldsymbol{k}_0 + \boldsymbol{k}_1 \begin{bmatrix} S_1 \\ \vdots \\ S_{N_\ell} \end{bmatrix} + \boldsymbol{k}_2 \begin{bmatrix} f(S_1, c_1) \\ \vdots \\ f(S_{N_\ell}, c_{N_\ell}) \end{bmatrix}, \tag{13a}$$

$$\boldsymbol{R}_c(\boldsymbol{S}, \boldsymbol{c}) = \boldsymbol{k}_3 + \boldsymbol{k}_4 \begin{bmatrix} S_1 c_1 \\ \vdots \\ S_{N_\ell} c_{N_\ell} \end{bmatrix} + \boldsymbol{k}_5 \begin{bmatrix} a(c_1) \\ \vdots \\ a(c_{N_\ell}) \end{bmatrix} + \boldsymbol{k}_6 \begin{bmatrix} m(c_1)c_1 f(S_1, c_1) \\ \vdots \\ m(c_{N_\ell})c_{N_\ell} f(S_{N_\ell}, c_{N_\ell}) \end{bmatrix} \tag{13b}$$

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. 12. 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

$$0 = R_{w,i}^k(S, c) = R_{w,i}(S_1^k, \ldots, S_{i-1}^k,\ S,\ S_{i+1}^{k-1}, S_{N_\ell}^{k-1},\ c_1^k, \ldots, c_{i-1}^k,\ c,\ c_{i+1}^{k-1}, c_{N_\ell}^{k-1}), \tag{14a}$$

$$0 = R_{c,i}^k(S, c) = R_{c,i}(S_1^k, \ldots, S_{i-1}^k,\ S,\ S_{i+1}^{k-1}, S_{N_\ell}^{k-1},\ c_1^k, \ldots, c_{i-1}^k,\ c,\ c_{i+1}^{k-1}, c_{N_\ell}^{k-1}). \tag{14b}$$

The single-cell problem of Eq. 14 takes the form of Eq. 12 and can be solved in the same robust manner that is 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, ..., N_B
for ℓ = 1, ..., N_B do
    if block B_ℓ contains only one cell then
        Solve single-cell problem Eq. 12 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, ..., N_ℓ do
                Solve single-cell problem Eq. 14 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. 9, to be able to utilize the reordering methods discussed in the two previous subsections. This operator splitting method was first introduced within streamline simulation [4, 10, 11], but can also offer certain benefits for finite-volume methods, e.g., as discussed in [14].

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

$$\big(b\phi S^*\big)_i^{n+1} - \big(b\phi S\big)_i^n + \frac{\Delta t}{V_i} \sum_j \big(bf(S^*, c^*)v\big)_{ij}^{n+1} = 0, \tag{15a}$$

$$\Big(b\phi c^* S^* + (1 - \phi_{\text{ref}})a(c^*)\Big)_i^{n+1} - \Big(b\phi cS + (1 - \phi_{\text{ref}})a(c)\Big)_i^n + \frac{\Delta t}{V_i} \sum_j \big(bm(c^*)c^* f(S^*, c^*)v\big)_{ij}^{n+1} = 0, \tag{15b}$$

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

$$\big(b\phi(S-S^*)\big)_i^{n+1} + \frac{\Delta t}{V_i}\sum_j \big(b\lambda_o f(S,c)(\rho_w-\rho_o)gK\nabla z\big)_{ij}^{n+1} = 0, \tag{16a}$$

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

In the segregation equation, all cells will in principle be coupled and hence solved for simultaneously using a Newton–Raphson method. However, in cases where the geo-cellular model only contain vertical cell columns, the segregation equations will decouple between columns and can hence 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 efficiently using the Thomas algorithm. Likewise, our 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. For cases without vertical columns, one may also foresee a nonlinear Gauss–Seidel method that iterates over a loop of column solves.

Finally, we note that by summing Eq. 15 and Eq. 16 we obtain

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

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

which implies, after summing over all the cells, that

$$\sum_i \big(b\phi S\big)_i^{n+1} = \sum_i \big(b\phi S\big)_i^{n}, \qquad \text{and} \qquad \sum_i \big(b\phi cS + (1-\phi_{\text{ref}})a(c)\big)_i^{n+1} = \sum_i \big(b\phi cS + (1-\phi_{\text{ref}})a(c)\big)_i^{n}. \tag{18}$$

In other words, the total masses of water, oil, and polymer are conserved.

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 [19]. 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 standard Newton–Raphson solver with the nested bracketing method as a fallback if Newton iterations fail.

4.1 Example 1: Polymer slug in a 1D reservoir. In the first example, we will verify our simulation against a commercial simulator [20] for three different polymer models of increasing complexity. To this end, we consider a 1D homogeneous reservoir 1500 m long 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. 3** show excellent agreement for all three cases, with only a slight deviation for the third model.

4.2 Example 2: a cubic reservoir 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. 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.

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	$m(c)$
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	$1.25 \cdot 10^{-7}$

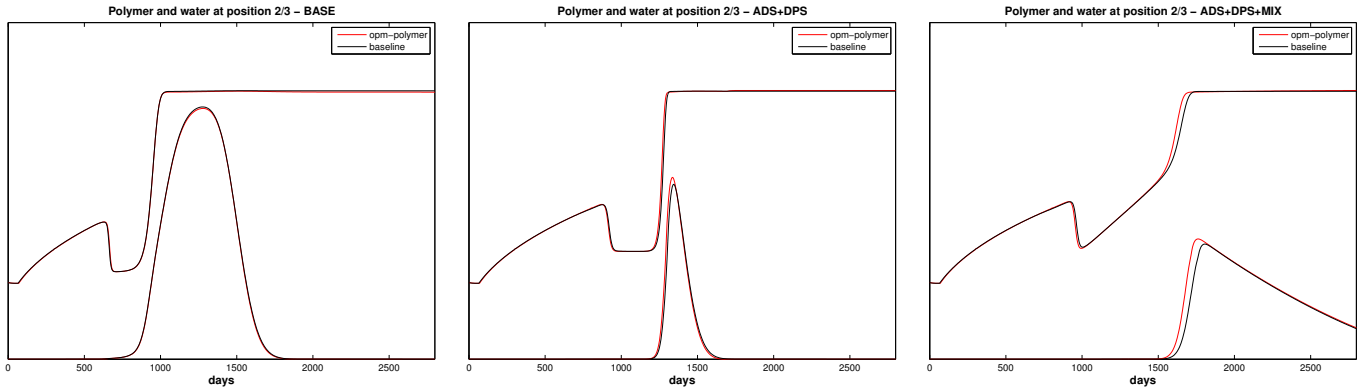

Fig. 3—Comparison of 1D polymer slug computed by the OPM polymer solver (red lines) and a commercial simulator (black lines) for three different model configurations: base-case model, base-case model with adsorption and dead pore space, and with mixing parameter $\omega = 0.5$.

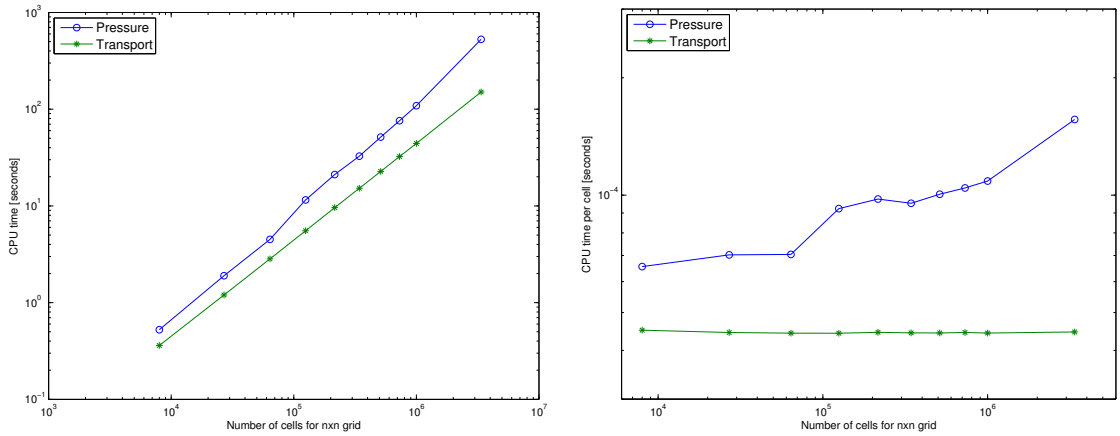

Fig. 4—CPU times for the $n \times n \times n$ reservoir in Example 2. The left plot shows total CPU time as a function of the size of the model. The right plot shows the CPU time per cell as a function of the size of the model.

Fig. 4 reports the CPU times for the pressure and transport solvers. The pressure solver uses the algebraic multigrid linear solver from `dune:istl` [3]. 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: Norne. 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 [12] 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 the view expressed in this example are the views of the authors and do not necessarily reflect the views of Statoil and the Norne license partners.

The Norne model shown in **Fig. 5** 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 6 neighbours, but some have as many as 21, due to faults. The permeability ranges from 0.3 mD to 4 darcies. The model contains 20 wells, of which 6 are injectors and 14 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 2** and Fig. 5, 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 $\omega = 1$.

TABLE 2—Fluid data for Example 3.

Viscosity	
c	$m(c)$
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

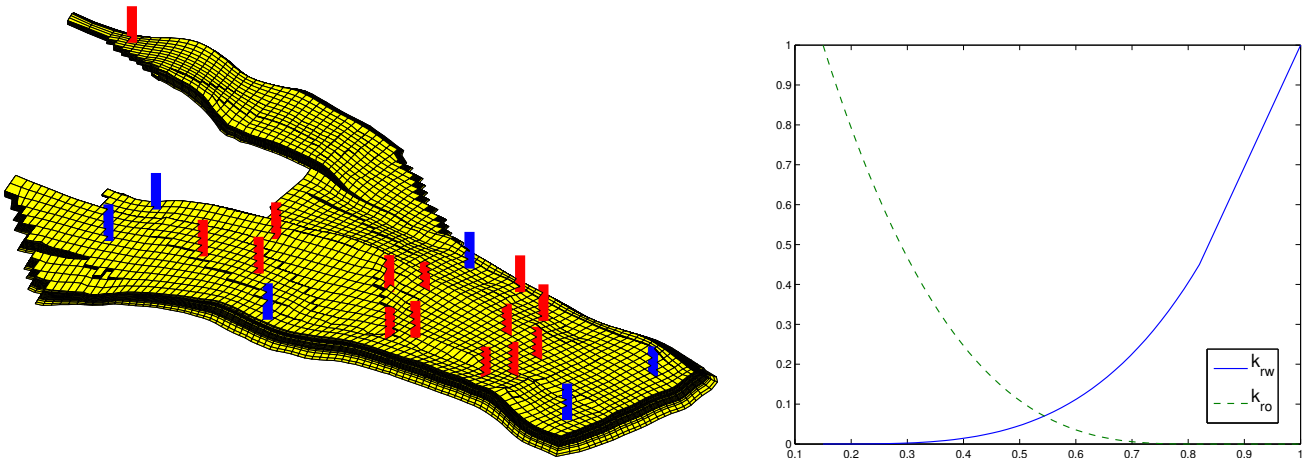

Fig. 5—Left plot: the Norne model with injection wells shown in blue and production wells shown in red. Right plot: relative permeability curves.

We compare a plain water injection scenario lasting 4000 days to a polymer injection scenario, in which a polymer solution 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. In the left part of **Fig. 6** we have plotted the total water cut of both scenarios, and can clearly see that the polymer has the effect of delaying water breakthrough, and changes the shape of the curves. In the right part we have plotted the water-cut curves of the three producers with earliest breakthrough.

The CPU time of the two simulations are reported in **Table 3**, 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

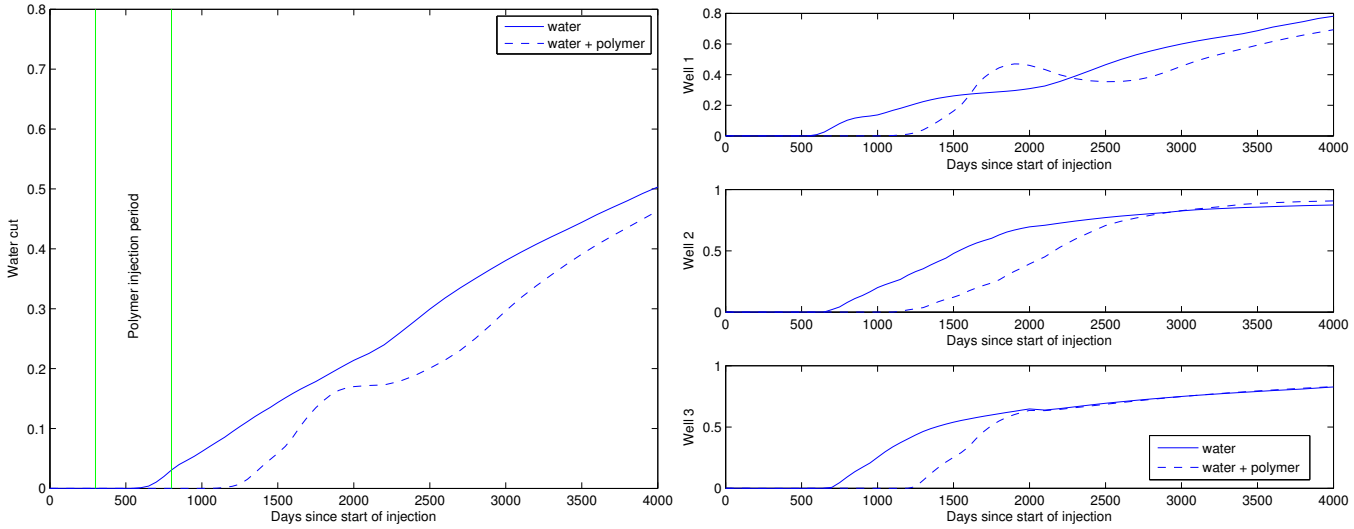

Fig. 6—Comparison of water-cut curves for Norne test case with water injection and polymer flooding.

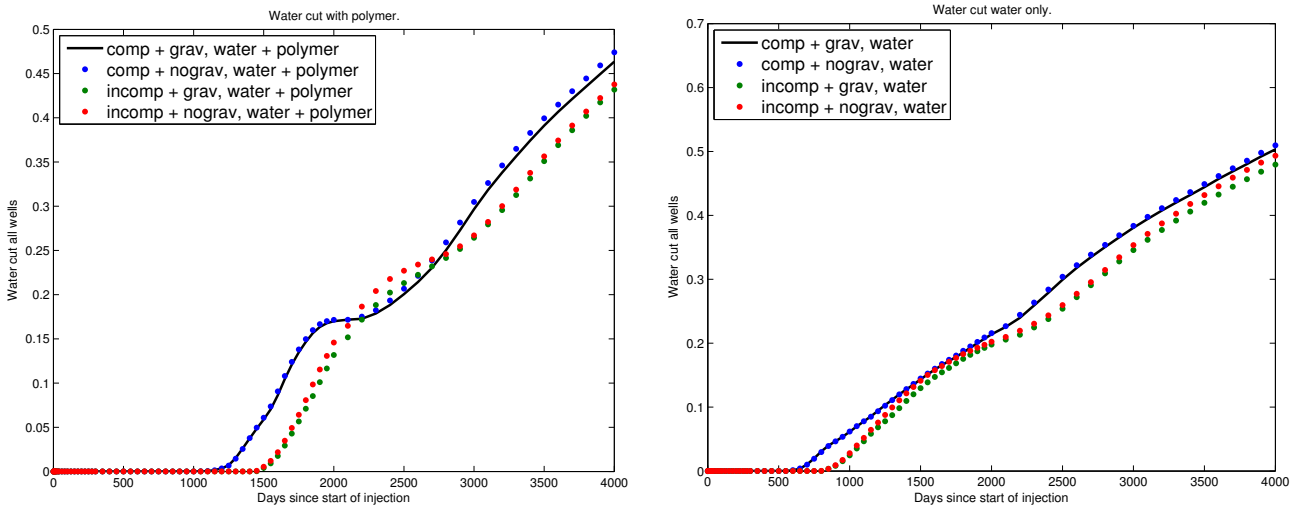

Fig. 7—Comparison of water-cut curves for the base-case model and models without gravity and/or compressibility for polymer flooding (left) and water injection (right).

reduces the runtime by a factor 2–3 in both cases. The effect of gravity is small in this case, the density difference between the two phases being 12%, and **Fig. 7** 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. Running with incompressible rock and fluids reduces runtime by a factor 3.7 for the pressure solver. 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 part.

TABLE 3—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

4.4 Example 4: a real field model. 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. 8 shows water saturation and polymer concentration after 1000 days. To compute the solution, we use two different time step sizes of 5 and 100 days, respectively, and in the advection step, we use both the bracketed and the Newton single-cell solver. Fig. 8 reports the corresponding number of iterations used in the last time step for all methods. A key observation from the figure, is that the computational work is limited to regions around wells and near the oil-water contact, where either the water saturation or polymer concentration changes.

For the pressure step, we use AGMG [1, 18] as linear solver. Each pressure solve converges within 4 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, thereby giving us the possibility of reducing the time-steps to increase the resolution of polymer fronts. 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 Conclusions

We have presented a highly efficient simulation method for realistic models of polymer flooding in heavy-oil reservoirs. The key points to the efficiency of our 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. A further advantage is that we have an unconditionally stable method for solving the localized single-cell problems that avoids time-step restrictions induced by stability problems. In particular, the method is very robust with respect to time steps and the large differences in cell volumes that are typically present in realistic high-resolution geo-cellular models. The method 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 method is implemented as part of the Open Porous Media (OPM) initiative [19] and is freely available from the website under the GPLv3 license. The current software implements a polymer model that includes dead pore space, adsorption, and permeability reduction. Initial testing indicate that the method works well even if the fluids have different compressibility, but further research is needed to understand the limitations for practical reservoir simulations. In addition, research will also be needed to develop efficient localization strategies for the nonlinear iterations for flow fields that contain large loops and have stronger gravity effects.

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

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

[2] A. M. AlSofi and M. J. Blunt. Streamline-based simulation of non-Newtonian polymer flooding. *SPE J.*, 15(4):895–905, 2010. doi: 10.2118/123971-PA.

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

[4] F. Bratvedt, T. Gimse, and C. Tegnander. Streamline computations for porous media flow including gravity. *Transp. Porous Media*, 25(1): 63–78, oct 1996. doi: 10.1007/BF00141262.

[5] T. Clemens, J. Abdev, and M. R. Thiele. Improved polymer-flood management using streamlines. *SPE J.*, 14(2):171–181, 2011. doi: 10.2118/132774-PA.

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

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

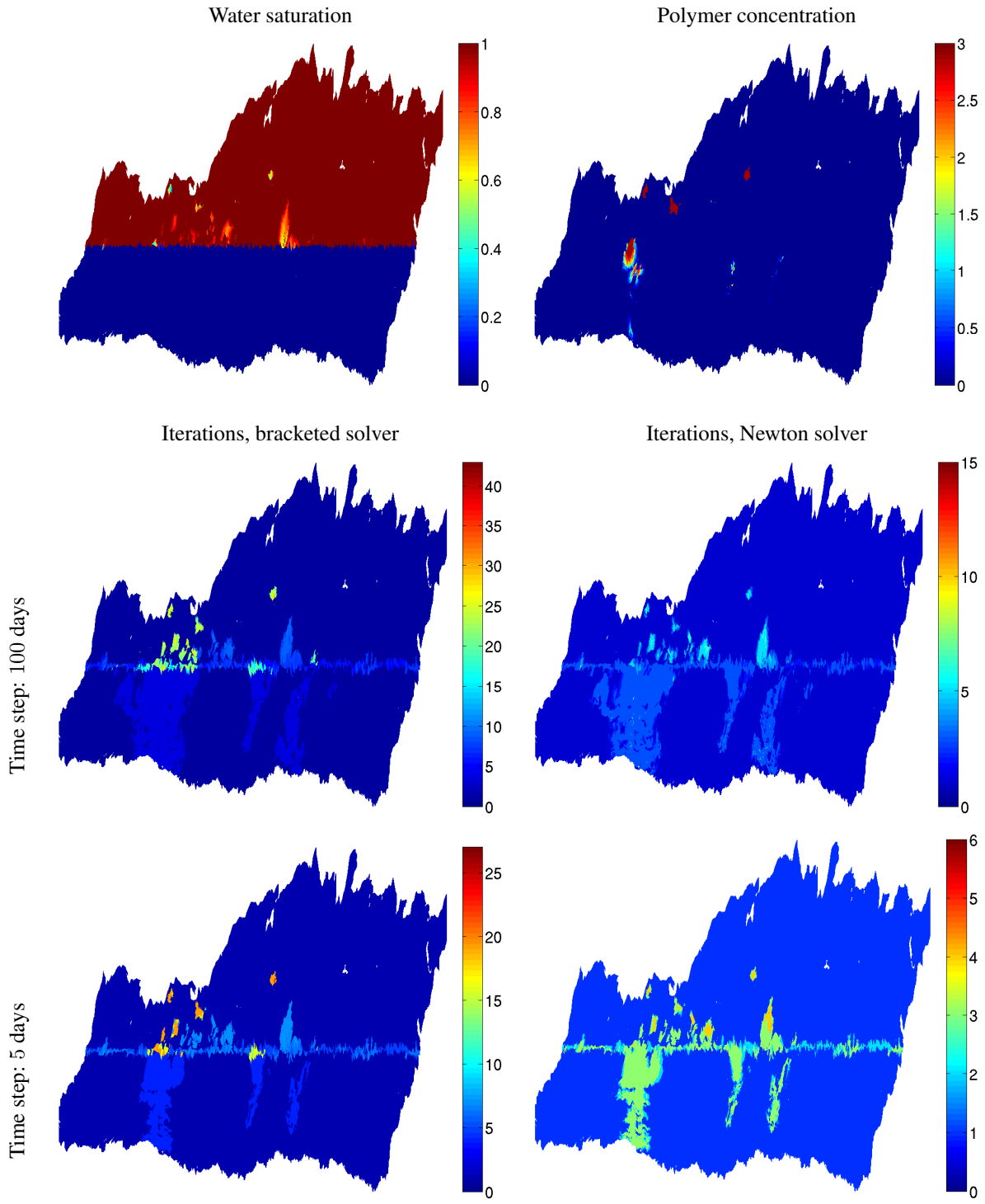

Fig. 8—Simulation of polymer injection for a real-field model. All plots show quantities from the last step in a simulation of 1000 days.

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

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

[10] R. H. J. Gmelig Meyling. A characteristic finite element method for solving non-linear convection-diffusion equations on locally refined grids. In D. Guerillot and O. Guillon, editors, *2nd European Conference on the Mathematics of Oil Recovery*, pages 255–262, Arles, France, Sept 11-14 1990. Editions Technip.

[11] R. H. J. Gmelig Meyling. 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, 1991. Studentlitteratur.

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

[13] F. Kwok and H. Tchelepi. Potential-based reduced Newton algorithm for nonlinear multiphase flow in porous media. *J. Comput. Phys.*, 227(1):706–727, 2007. doi: 10.1016/j.jcp.2007.08.012.

[14] K.-A. Lie, J. R. Natvig, and H. M. Nilsen. 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, 2012.

[15] K.-A. Lie, H. M. Nilsen, A. F. Rasmussen, and X. Raynaud. 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, 2012.

[16] J. R. Natvig and K.-A. Lie. 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, 2008. ISSN 0021-9991. doi: 10.1016/j.jcp.2008.08.024. URL `http://dx.doi.org/10.1016/j.jcp.2008.08.024`.

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

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

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

[20] Eclipse. *Eclipse Technical Description Manual*. Schlumberger, 2009.2 edition, 2009.

[21] M. R. Thiele, R. P. Batycky, S. Pöllitzer, and T. Clemens. Polymer-flood modeling using streamlines. *SPE J.*, 13(2):313–322, 2010. doi: 10.2118/115545-PA.

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

[23] F. R. Wassmuth, K. Green, W. Arnold, and N. Cameron. Polymer flood application to improve heavy oil recovery at East Bodo. *J. Can. Petrol. Technol.*, 48(2):55–61, 2009. doi: 10.2118/09-02-550.

[24] K. Xiaodong, Z. Jian, S. Fujie, Z. Fengjiu, F. Guozhi, Y. Junru, Z. Xiansong, and X. Wentao. 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*, 2011.