Non-linear Newton Solver for a Polymer Two-phase System Using Interface-localized Trust Regions

Øystein Klemetsdal Olav Møyner Xavier Raynaud Knut-Andreas Lie

March 4, 2017

Abstract

Models of polymer flooding account for several processes such as concentration dependent viscosity, adsorption, incomplete mixing, inaccessible pore space, and reduced permeability effects, which altogether gives strongly coupled nonlinear systems that are challenging to solve numerically. Herein, we use a sequentially implicit solution strategy that splits the equation system into a pressure and a transport part. Our objective is to improve convergence rates for the transport subproblem, which contains many of the essential nonlinearities caused by the addition of polymer. Convergence failure for the Newton solver is usually caused by steps that pass inflection points and discontinuities in the fractional flow functions. The industry-standard approach is to heuristically chop time steps and/or dampen saturation updates suggested by the Newton solver if these exceed a predefined limit. An improved strategy is to use trust regions to determine safe saturation updates that stay within regions having the same curvature for the numerical flux. This approach has previously been used to obtain unconditional convergence for waterflooding scenarios and multicomponent problems with realistic property curves. Herein, we extend the method to polymer flooding, and study the performance of the method for a wide range of polymer parameters and reservoir configurations.

Introduction

Injecting water to maintain pressure and displace and move oil towards producing wells is a common strategy for improving recovery from onshore and offshore oil fields. If oil is more viscous than the injected water, large volumes of oil are not recoverable by water injection alone, as the unstable displacement can redirect the injected water through high-permeable zones or spontaneously induced viscous fingers. In such cases, a diluted solution of long-chain polymer molecules can be added to the injected water to increase its viscosity, which in turn improves local displacement and sweep efficiency. However, capital and operational expenditures cannot be justified until the efficiency of polymer flooding is proven on a field scale. The business case of EOR operations is typically demonstrated through reservoir simulation.

Reservoir simulators used in industry are predominately based on a black-oil formulation with simple first-order, upstream-mobility weighting for spatial discretization, and fully implicit time stepping. This offers unconditional stability for a wide range of reservoir heterogeneities and physical flow regimes in water and gas flooding. Numerical simulation of polymer flooding is significantly more challenging than traditional waterflooding for several reasons: The diluted polymer solution is (partially) miscible with water and tends to form a complex fingering pattern induced by media heterogeneity. On top of this, there are complex and adverse phenomena like adsorption onto the rock, cross-link polymer plugging, degradation and in-situ chemical reactions, dead pore space, and non-Newton fluid rheology that may strongly affect injectivity. In its simplest form, polymer flooding is described as an immiscible, two-phase, three-component fluid system in which an empirical mixture model is used to account for unresolved miscibility effects. Most models will also include the effect of adsorption and/or permeability and porosity reduction. Compared with the basic model for waterflooding, the polymer model will have an extra conservation equation for polymer mass and new constitutive relationships. These introduce stronger nonlinearities in the accumulation and flux terms of the flow model and make the accuracy of the solution more sensitive to grid resolution. A potential for polymer flooding can often be shown in small-scale or sector models. On full-field models, on the other hand, numerical diffusion tends to mask the effect of polymer unless the grid resolution is significantly increased compared with cell sizes that are sufficient to simulate waterflooding. This makes simulation of polymer flooding using conventional serial simulators prohibitively costly.

One alternative to reduce computational time is through parallelization, e.g., as recently demonstrated by Petoro for the Johan Sverdrup field (Nasiri and Henriquez, 2014). Other approaches include dynamic gridding (see e.g., Hoteit and Chawathé (2016)) and streamline simulation (AlSofi and Blunt, 2010; Clemens et al., 2011). The computational efficiency of streamline simulation comes mainly from exploiting weak couplings in the flow equations to formulate sequential solution procedures with large time-step capabilities and using efficient solvers to compute the transport of fluid phases and chemical components. For several years, our group has been developing a related approach which is similar to streamline simulation in that it relies on a sequential formulation to decouple the computation of pressure and fluid transport, and exploits certain uni-directional flow properties to formulate fast transport solvers. However, unlike in streamline simulation, we do not introduce a new 1D Lagrangian grid system to discretize the transport equations, but rather use unidirectional flow properties to devise localized and highly efficient nonlinear solvers for transport equations discretized using a standard finite-volume method. This idea was first introduced by Natvig and Lie (2008), and has later been developed for polymer flooding (Lie et al., 2012, 2014). In the best case, these nonlinear solvers can loop through all cells in the reservoir in a particular order determined by the flow directions of the problem and update saturations and polymer concentrations one cell at the time by solving single-cell problems using unconditionally stable, bracketed root-finding methods. More generally, the ordering will consist of cells with co-current flow, which can be solved individually, and cells that contain countercurrent flow and thus must be solved for simultaneously, e.g., using the standard Newton–Raphson method or a nonlinear Gauss–Seidel method. Recently, we also introduced a multiscale solver for accelerating the pressure update (Hilden et al., 2016).

The combination of highly nonlinear equations, orders-of-magnitude variation in petrophysical prop-

erties, and fine grid cells with non-Cartesian cell geometries and high aspect ratios can pose severe restrictions on time-step lengths. For this reason, it is important to develop fast, accurate and robust solvers tailored specifically towards enhanced oil recovery (EOR) problems. In this paper, our focus is on improving robustness by reducing the number of convergence failures in the transport solver. Using *trust regions* with Newton-Raphson's method to select safe updates and guarantee convergence (Jenny et al., 2009) has proven successful for various two-phase flow scenarios, including buoyancy effects (Wang and Tchelepi, 2013), capillary forces (Li and Tchelepi, 2014), and compositional phase behavior (Voskov and Tchelepi, 2011) with analytical relative permeability curves. More recently, the method was extended to multiphase problems with general property curves by Møyner (2016) based on an approximate reconstruction of the flux function along the Newton path. In this work, we extend this idea to EOR problems exemplified by polymer injection. For simplicity, we only discuss simultaneous updates of all cells in the grid, but the reader should keep in mind that our new algorithm can easily be made even faster by localizing the Newton updates to cells that are part of regions containing counter-current flow, and using an optimal cell-by-cell update in regions of co-current flow.

Model equations

The starting point for the polymer model studied herein is the standard black-oil model for an oil–water system, which consists of two mass-conservation equations. By introducing a backward Euler temporal discretization, we write these on semi-discrete residual form for time-step $n+1$

$$R_\alpha = \frac{1}{\Delta t}\left[(\phi\rho_\alpha S_\alpha)^{n+1} - (\phi\rho_\alpha S_\alpha)^n\right] + \nabla\cdot(\rho_\alpha \vec{v}_\alpha)^{n+1} - (\rho_\alpha q_\alpha)^{n+1} = 0, \qquad \alpha \in \{w,o\}, \tag{1}$$

$$\vec{v}_\alpha = -\frac{k_{r\alpha}}{\mu_\alpha}\boldsymbol{K}(\nabla p_\alpha - \rho_\alpha g \nabla z). \tag{2}$$

Here, ρ_α, p_α, $\vec{v}_\alpha$, S_α, and μ_α denote density, pressure, Darcy velocity, saturation, and viscosity of phase α; ϕ and $\boldsymbol{K}$ are the porosity and permeability of the rock; the relative permeability $k_{r\alpha}$ models reduced permeability for one phase in the presence of the other; whereas g denotes gravity acceleration, z the vertical coordinate, and q_α volumetric sources and sinks (wells). To close the model, we assume that the phases fill the pore space completely, $S_o + S_w = 1$ and that the phase pressures are related through a saturation-dependent capillary pressure, $p_o - p_w = P_c(S_w)$. The relationship between density and pressure is modelled using inverse formation-volume-factors, $\rho_\alpha = b_\alpha(S_\alpha)\rho_\alpha^s$, where ρ_α^s is the constant surface density of phase α. A similar relationship, $\phi = b_r\phi_0$, is introduced to model rock compressibility. We pick oil pressure and water saturation as primary unknowns, henceforth denoted p and S, respectively. Capillary-pressure terms are omitted herein to simplify derivations, but are included in our simulator.

The polymer system

To model diluted polymer, we introduce an additional conservation equation, written on semi-discrete residual form

$$R_p = \frac{(\phi b_w c S)^{n+1} - (\phi b_w c S)^n}{\Delta t} + \rho_r(1-\phi_0)\frac{c_a^{n+1} - c_a^n}{\Delta t} + \nabla\cdot(c b_w \vec{v}_p)^{n+1} - (b_w q_p)^{n+1} = 0. \tag{3}$$

Here, $c \in [0, c^*]$ denotes polymer concentration, c_a is the adsorption concentration which models polymer molecules attached to the rock surface as a function of c, and ρ_r denotes rock density. Diluted polymer and water form a miscible system, but will here be modelled as an immiscible system by introducing effective mixture viscosities $\mu_{w,\text{eff}}$ and $\mu_{p,\text{eff}}$ that depend on polymer concentration. Polymer adsorption and polymer molecules lodged in narrow pore throats will both contribute to reduce the permeability experienced by the water–polymer mixture. Altogether, these effects are accounted for by the following modified Darcy equations

$$\vec{v}_w = -\frac{k_{rw}(S)\boldsymbol{K}}{\mu_{w,\text{eff}}(c)R_k(c)}(\nabla p - \rho_w g\nabla z), \qquad \vec{v}_p = \frac{\mu_{w,\text{eff}}}{\mu_{p,\text{eff}}}\vec{v}_w = m(c)\vec{v}_w. \tag{4}$$

Adsorption c_a and permeability reduction R_k will both also generally depend on the maximum concentration observed over the simulation history, which introduces hysteretic behavior in the model. The

viscosity of the water–polymer mixture is given by the Todd and Longstaff (1972) model, which uses a parameter $\omega \in [0,1]$ to describe the degree to which polymer mixes into the aqueous phase, from no mixing for $\omega = 0$ to full mixing for $\omega = 1$,

$$\frac{1}{\mu_{w,\text{eff}}} = \frac{1-c/c^*}{\mu_m(c)^\omega \mu_w^{1-\omega}} + \frac{c/c^*}{\mu_m(c)^\omega \mu_m(c^*)^{1-\omega}}, \qquad m(c) = \left[\left(1-\frac{c}{c^*}\right)\left(\frac{\mu_m(c^*)}{\mu_w}\right)^{1-\omega} + \frac{c}{c^*}\right]^{-1}. \tag{5}$$

Here, $\mu_m(c)$ is the viscosity of a fully mixed system and $m(c)$ is the ratio between the effective polymer and water viscosities. The mixing parameter ω depends on the heterogeneity and the flow pattern.

Sequential splitting: pressure and transport equations

One approach for solving (1)–(3) is to reformulate the system into a pressure equation and system of two transport equations, and then solve the two parts sequentially. We first define the pressure equation as a linear combination of the individual residual equations (1) weighted by $1/b_\alpha^{n+1}$, which enables us to eliminate the saturations at the end of the time step using the relation $S_w + S_o = 1$,

$$R_{pres} = \frac{\phi^{n+1}}{\Delta t} - \left[\frac{b_w^n S_w^n}{b_w^{n+1}} + \frac{b_o^n S_o^n}{b_o^{n+1}}\right]\frac{\phi^n}{\Delta t} + \frac{\nabla\cdot(b_w\vec{v}_w)^{n+1}}{b_w^{n+1}} + \frac{\nabla\cdot(b_o\vec{v}_o)^{n+1}}{b_o^{n+1}} - (q_w+q_o)^{n+1} = 0. \tag{6}$$

Note that whereas the polymer equation does not factor into the pressure equation directly, the effect of polymer upon the water viscosity is accounted for. Likewise, we can reformulate the conservation equations by introducing the total Darcy velocity defined as the sum of the individual phase velocities $\vec{v} = \vec{v}_w + \vec{v}_o$. We have

$$\vec{v} = -(\lambda_w+\lambda_o)\boldsymbol{K}\nabla p + g(\lambda_w\rho_w + \lambda_o\rho_o)\boldsymbol{K}\nabla z. \tag{7}$$

Here, we have introduced the phase mobility $\lambda_\alpha = k_{r\alpha}/\mu_\alpha$ (with $\mu_{w,\text{eff}}$ for the water phase). If we further define the fractional flow function $f_\alpha = \lambda_\alpha/(\lambda_w+\lambda_o)$, the flux function for each phase reads,

$$\vec{v}_w = f_w[\vec{v} + \lambda_o(\rho_w-\rho_o)\boldsymbol{K}g\nabla z] = f_w(\vec{v}+\vec{G}_w), \tag{8}$$

$$\vec{v}_o = f_o[\vec{v} + \lambda_w(\rho_o-\rho_w)\boldsymbol{K}g\nabla z] = f_o(\vec{v}+\vec{G}_o). \tag{9}$$

Since we have new and equivalent expressions for the fluxes, we can write the transport equations,

$$R_\alpha = \frac{(\phi b_\alpha S_\alpha)^{n+1} - (\phi b_\alpha S_\alpha)^n}{\Delta t} + \nabla\cdot\left[b_\alpha f_\alpha(\vec{v}+\vec{G}_\alpha)\right]^{n+1} - (b_\alpha q_\alpha)^{n+1} = 0, \qquad \alpha \in \{w,o\},$$

$$R_p = \frac{(\phi b_w c S)^{n+1} - (\phi b_w c S)^n}{\Delta t} + \rho_r(1-\phi_0)\frac{c_a^{n+1}-c_a^n}{\Delta t} + \nabla\cdot\left[m(c)cb_w f_w(\vec{v}+\vec{G}_w)\right]^{n+1} - (b_w q_p)^{n+1} = 0.$$

If we now solve the pressure equation with fixed saturation and polymer concentration, we obtain pressure and total velocity. We can then use the water and polymer equations with fractional flow flux functions to advance saturation and polymer concentration a period Δt forward in time. Afterwards, the oil saturation is found from the closure relation $S_o = 1 - S_w$. This procedure is repeated until we reach the desired time horizon.

Finite-volume discretization

To derive a fully discrete model, we introduce a grid consisting of cells C_i with a bulk volume V_i and integrate over each cell in space, using first the divergence theorem and then the midpoint rule to evaluate surface integrals,

$$R_{\alpha,i} = \left[(\phi b_\alpha S_\alpha)_i^{n+1} - (\phi b_\alpha S_\alpha)_i^n\right] + \frac{\Delta t}{V_i}\sum_j |\Gamma_{ij}|(b_\alpha\vec{v}_\alpha\cdot\vec{n})_{ij}^{n+1} - \frac{\Delta t}{V_i}(b_\alpha q_\alpha)_i^{n+1} = 0, \quad \alpha\in\{w,o\},$$

$$R_{p,i} = \left[(\phi b_w cS)_i^{n+1} - (\phi b_w cS)_i^n\right] + \rho_r(1-\phi_0)\left[c_{a,i}^{n+1} - c_{a,i}^n\right] + \frac{\Delta t}{V_i}\sum_j |\Gamma_{ij}|(m(c)cb_w\vec{v}_w\cdot\vec{n})_{ij}^{n+1} - \frac{\Delta t}{V_i}(b_w q_p)_i^{n+1} = 0. \tag{10}$$

Here, the double subscript ij denotes quantities evaluated at the interface Γ_{ij} between cells C_i and C_j, $|\Gamma_{ij}|$ is the corresponding interface area and $\vec{n}$ the surface normal. Hence, the pressure equation reads,

$$\phi_i^{n+1} - \sum_{\alpha=w,o} \frac{(\phi b_\alpha S_\alpha)_i^n}{b_{\alpha,i}^{n+1}} + \frac{\Delta t}{V_i} \sum_j \sum_{\alpha=w,o} \frac{|\Gamma_{ij}|}{b_{\alpha,i}^{n+1}} \left(b_\alpha \vec{v}_\alpha \cdot \vec{n}\right)_{ij}^{n+1} - \frac{\Delta t}{V_i} (q_w + q_o)_i^{n+1} = 0. \tag{11}$$

Using a two-point approximation, we can expand the interface fluxes for water and oil as follows,

$$(b_\alpha v_\alpha)_{ij} = |\Gamma_{ij}| \left(b_\alpha \vec{v}_\alpha \cdot \vec{n}\right)_{ij} \approx \underbrace{(b_\alpha \lambda_\alpha)_{ij}}_{\text{upstream evaluation}} \left[T_{ij}(p_i - p_j) - \underbrace{(b_\alpha)_{ij}}_{\text{arithmetic average}} \rho_\alpha^s g T_{ij}(z_i - z_j) \right]. \tag{12}$$

Here, the transmissibility T_{ij} between cells C_i and C_j is defined as

$$T_{ij} = \left[T_{i,j}^{-1} + T_{j,i}^{-1} \right]^{-1}, \qquad T_{i,j} = |\Gamma_{ij}| \boldsymbol{K}_i \frac{\vec{c}_{i,j} \cdot \vec{n}_{i,j}}{|\vec{c}_{i,j}|^2}, \tag{13}$$

where $\vec{n}_{ij}$ is the normal vector to Γ_{ij} and $\vec{c}_{i,j}$ is the vector from the centroid of C_i to the centroid of Γ_{ij}.

In the sequential solution strategy, the saturation and concentrations are held fixed when solving the nonlinear pressure equation (11). Hence, all saturation and concentration dependencies in (14) are evaluated using values from the end of the previous time step and upstream directions are determined using values from the previous iteration of the Newton–Raphson solver. Once the pressure iteration is converged, the fluxes v_{ij} corresponding to the total velocity (7) are computed in the same way as in (14) and are held fixed during the transport step. The phase fluxes are then approximated as

$$v_{\alpha,ij} = |\Gamma_{ij}| \left(\vec{v}_\alpha \cdot \vec{n}\right)_{ij} \approx f_{\alpha,ij} \left[v_{ij} + \sum_{\beta \neq \alpha} \lambda_{\alpha,ij} (\rho_\alpha - \rho_\beta) g T_{ij} (z_i - z_j) \right] \tag{14}$$

The phase mobilities $\lambda_{\alpha,ij}$ and the fractional flow functions $f_{\alpha,ij}$ are evaluated from the upstream direction as follows

$$\lambda_{\alpha.ij} = \begin{cases} \lambda_{\alpha,i} & \text{if } v_{\alpha,ij} > 0, \\ \lambda_{\alpha,j} & \text{if } v_{\alpha,ij} < 0. \end{cases} \tag{15}$$

Note that these definitions are implicit. Explicit formulations have been derived by Brenier and Jaffré (1991) and are used in our implementation.

Trust-region algorithm

We introduce a short hand for the primary variables of the transport equations, $\boldsymbol{\xi} = (S_1, \ldots, S_n, c_1, \ldots, c_n)$, and analogously for the residual equations $\boldsymbol{R} = (R_{w,1}, \ldots, R_{w,n}, R_{p,1}, \ldots, R_{p,n})$. For brevity, we also drop superscripts $n+1$. Solving the transport equations for a time step Δt consists of finding $\boldsymbol{\xi}$ such that $\boldsymbol{R}(\boldsymbol{\xi}) = \boldsymbol{0}$. We rewrite the residuals for the transport equations as

$$R_{\gamma,i}(\boldsymbol{\xi}) = A_{\gamma,i}(\boldsymbol{\xi}_i) + \sum_j F_{\gamma,ij}(\boldsymbol{\xi}_i, \boldsymbol{\xi}_j) - \frac{\Delta t}{V_i} (b_w q_\gamma)_i^{n+1} = 0, \quad \gamma \in \{w, p\}.$$

where the definitions of the accumulation terms $A_{\gamma,i}$ and the flux functions $F_{\gamma,ij}$ follow from (10), and $\boldsymbol{\xi}_i = (S_i, c_i)$. Note that $F_{\gamma,ij} = -F_{\gamma,ji}$. The linearized update to the primary variables in the Newton method is given by

$$\Delta \boldsymbol{\xi} = -\boldsymbol{J}(\boldsymbol{\xi})^{-1} \boldsymbol{R}(\boldsymbol{\xi}), \tag{16}$$

where $\boldsymbol{J}$ is the Jacobian matrix of $\boldsymbol{R}$. A full Newton update corresponds to setting $\boldsymbol{\xi}^{\ell+1} = \boldsymbol{\xi}^{\ell} + \Delta \boldsymbol{\xi}^{\ell}$, where ℓ is the iteration counter. It is well known that Newton's method will experience convergence issues whenever the update passes inflection points or kinks in $\boldsymbol{R}$. A standard approach is then to successively shorten the time step until the Newton solver manages to converge. The main drawback of this is that the nonlinear solver may waste a huge number of iterations before convergence. Another popular approach, introduced first by Jenny et al. (2009), is to limit the Newton step using a *trust region* defined

by kinks and inflection points in the residual function. The idea is to find damping factors $\theta_i \in [0,1]$ so that the updates do not pass far beyond such points:

$$\boldsymbol{\xi}_i^{\ell+1} = \boldsymbol{\xi}_i^{\ell} + \theta_i \Delta \boldsymbol{\xi}_i^{\ell}. \tag{17}$$

We consider the interface between cell i and cell j, and assume that we have found the Newton updates $\Delta\boldsymbol{\xi}_i = (\Delta S_i, \Delta c_i)$ and $\Delta\boldsymbol{\xi}_j = (\Delta S_j, \Delta c_j)$. Since this is a multi-component problem, it is natural to consider the direction of the Newton update defined as

$$\boldsymbol{d} = \frac{(\Delta\boldsymbol{\xi}_i, \Delta\boldsymbol{\xi}_j)}{\|(\Delta\boldsymbol{\xi}_i, \Delta\boldsymbol{\xi}_j)\|}. \tag{18}$$

By considering possible updates in this direction, the process of identifying the trust region reduces to a one-dimensional problem.

We now investigate possible inflection points and kinks in the residual functions. As explained by Møyner (2016), the water accumulation terms $A_{w,i}$ are linear, and will thus not cause convergence issues. This is not the case for the polymer accumulation terms, whose second-order derivatives read

$$\frac{\partial^2 A_{p,i}}{\partial S_j \partial S_k} = 0, \ \forall j,k, \qquad \frac{\partial^2 A_{p,i}}{\partial c_j \partial c_k} = \rho_r(1-\phi_0)c_a''(c_i)\delta_{ij}\delta_{ik}, \qquad \frac{\partial^2 A_{p,i}}{\partial S_j \partial c_k} = b_{w,i}\phi_i\delta_{ij}\delta_{ik}.$$

Here, δ_{ij} is the Kronecker delta. We see that the derivatives are either constant, or proportional to the second-order derivative of the adsorption function c_a. Herein, we model the adsorption function as

$$c_a = (1 - e^{-Dc})c_a^*,$$

for some decay parameter D and maximum adsorbed concentration c_a^*. The polymer accumulation term is thus free of inflection points and kinks. From these observations, we conclude that the major source of convergence issues for the nonlinear solver originates from the flux functions $F_{\gamma,ij}$. The directional derivatives of these functions read

$$\partial_{\boldsymbol{d}} F_{\gamma,ij} = \left(\nabla_{\boldsymbol{\xi}_i} F_{\gamma,ij}, \nabla_{\boldsymbol{\xi}_j} F_{\gamma,ij}\right) \cdot \boldsymbol{d}, \qquad \text{where} \qquad \nabla_{\boldsymbol{\xi}_k} F_{\gamma,ij} = \left(\frac{\partial F_{\gamma,ij}}{\partial S_k}, \frac{\partial F_{\gamma,ij}}{\partial c_k}\right),$$

with the update direction $\boldsymbol{d}$ defined in (18). Since we are interested in finding a trust region along the Newton update, we define $\boldsymbol{\xi}(\theta) = \boldsymbol{\xi}^{\ell} + \theta\Delta\boldsymbol{\xi}^{\ell}$, and write $F_\gamma'(\theta) = \partial_{\boldsymbol{d}} F_{\gamma,ij}(\boldsymbol{\xi}(\theta))$. Note that have omitted the interface subscripts ij in order to simplify the notation. In the same manner, we define the second-order derivative in the direction of the update: $F_\gamma''(\theta) = \partial_{\boldsymbol{d}}^2 F_{\gamma,ij}\big(\boldsymbol{\xi}(\theta)\big)$.

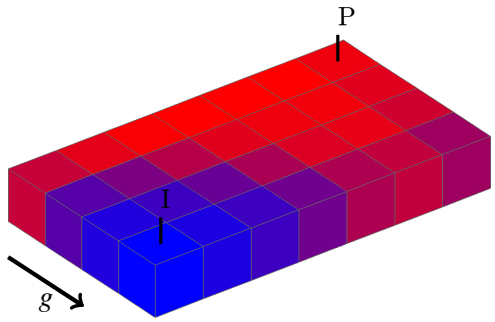
If we have an analytical expression for F_γ, we can proceed to find the inflection points and kinks of F_γ as a function of the damping factor θ using $F_\gamma''(\theta)$. However, since the second-order derivatives of the flux function is not always well defined—for instance in the case of piecewise linear relative permeabilities—and because they might vanish over large regions of saturation and concentration, we use a monotone interpolation scheme for F_γ'. Taking the derivative of this gives an approximation of F_γ''. With this information, we can determine a trust region for each flux function $F_{\gamma,ij}$, which we use to choose the largest possible safe update parameter $\theta_{\gamma,ij}$ for our nonlinear iteration. Ideally, this update ends just on the other side of an inflection point or kink.

Local and global chopping

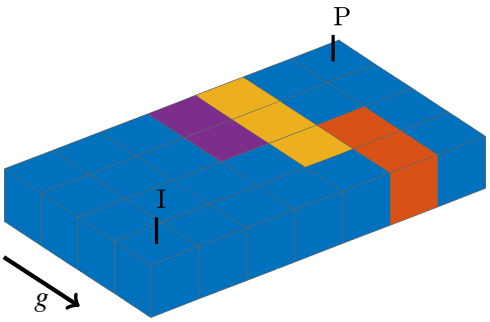
In the existing literature, it is common to set all trust-region parameters θ_i equal to the smallest of all trust-region interface parameters:

$$\theta = \min_{\gamma,i,j}\{\theta_{\gamma,ij}\}.$$

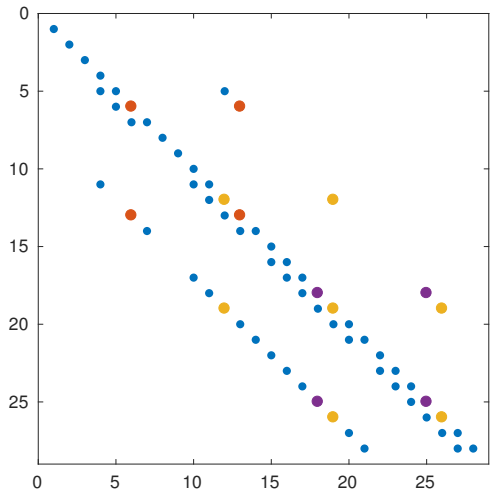
While this global chopping strategy offers unconditional convergence, and is the only approach guaranteed not to modify the Newton direction, it may be far too conservative in some cases. We will instead



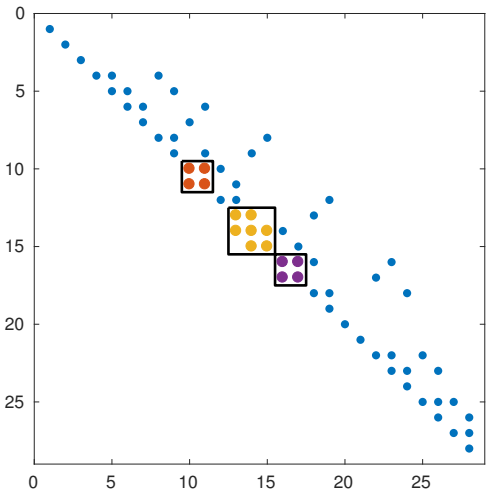
(a) Saturation profile.



(b) The three cycles, indicated by purple, yellow and red.



(c) Sparsity pattern. Cycle cells (diagonal) and cycle interfaces (off-diagonals) are indicated by larger dots.



(d) Sparsity pattern of reordered matrix.

Figure 1: Cycles and sparsity pattern of $\boldsymbol{C}_{\Delta\boldsymbol{\xi}}$.

follow the approach of Møyner (2016) and apply a local chopping procedure. This approach is based on the observation that the equations governing fluid transport in most cases have finite speed of propagation. In practice, this means that if we have multiple fronts propagating, these will not interact until a given time. The implication of this is that the Newton solver may converge faster in some parts of the domain than others. To exploit this property, we start by identifying cells that depend strongly on each other. To this end, we define a connectivity matrix $\boldsymbol{C}_{\Delta\boldsymbol{\xi}}$ by

$$\boldsymbol{C}_{\Delta\boldsymbol{\xi}} = \sum_{\gamma=w,p} \sum_{\eta=S,c} \boldsymbol{C}^{\gamma}_{\Delta\eta}, \quad \text{where} \quad \left(\boldsymbol{C}^{\gamma}_{\Delta\eta}\right)_{i,j} = \begin{cases} 1, & \text{if } \left|\frac{\partial R_{\gamma,i}}{\partial \eta_j}\Delta\eta_j\right| \geq \varepsilon \left|\frac{\partial R_{\gamma,i}}{\partial \eta_i}\Delta\eta_i\right|, \\ 0, & \text{otherwise.} \end{cases} \tag{19}$$

That is, the connection between cell i and cell j is strong if either the saturation or concentration update in cell j has an impact on the saturation or concentration error in cell i larger than a given threshold ε. The matrix $\boldsymbol{C}_{\Delta\boldsymbol{\xi}}$ is a directed graph, and the procedure of finding the global relaxation factors can then be summarized as follows:

1. Find all cycles in $\boldsymbol{C}_{\Delta\boldsymbol{\xi}}$, and combine all nodes in each cycle into a single (super)node.
2. Assign a relaxation factor to each cell by finding the smallest of the relaxation values of all interfaces with nonzero flux connected to that cell. Cells belonging to a cycle are then assigned the minimum value for all cells in that cycle.
3. Perform a topological sort of the resulting modified connection matrix without cycles.
4. Traverse the graph, and assign to each cell the minimum relaxation factor of itself and all cells upstream to it.

Figure 1 shows a simple example in which we inject a heavy fluid in one corner, and produce fluids in the opposite corner. We clearly see the formation of cycles at the saturation front.

Oscillation detection

While the trust-region algorithm ensures that the Newton solver always converges, it may be overly conservative, since it might reduce the update even when it is not necessary. This is illustrated in Figure 2, in which the initial guess and the solution are on the far opposite sides of an inflection point. In this case, taking the full Newton step is much faster than using the trust-region algorithm, as the initial linearized update steps over the inflection point, ending up (by chance) in the same contraction region as the final solution. We can utilize the fact that problematic inflection points, local minima, and kinks in the resid-

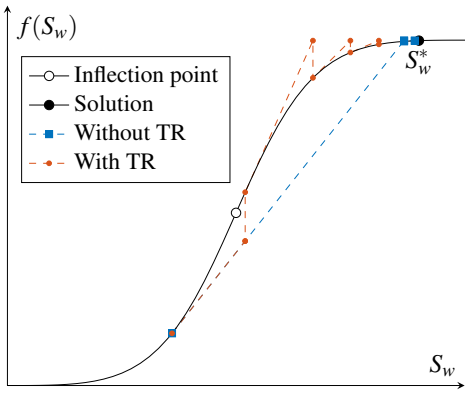


Figure 2: An example where the trust region algorithm is too conservative.

ual function can be identified by oscillations in the Newton updates. In particular, if the saturation or concentration update changes sign from one iteration to the next, we have either passed the solution, an inflection point, or a local minimum. Thus, to further improve our algorithm, we apply the trust-region check only when oscillations are present over an interface. We denote the updates in cell C_i at Newton iteration ℓ of time step n by $(\Delta S_i^{n_\ell}, \Delta c_i^{n_\ell})$, and say that we have oscillations over interface Γ_{ij} if

$$\text{sign}(\Delta S_l^{n_\ell}) \neq \text{sign}(\Delta S_l^{n_{\ell+1}}) \quad \text{and/or} \quad \text{sign}(\Delta c_l^{n_\ell}) \neq \text{sign}(\Delta c_l^{n_{\ell+1}}), \quad l \in \{i, j\},$$

and apply the trust region algorithm if this is the case. For large time steps, the initial updates will be large, and it is likely that we will pass problematic regions in the flux function. In such cases, we will not gain much by applying the above procedure. For shorter time steps, on the other hand, it will not always be necessary to apply the trust-region algorithm, since the updates will usually be relatively small. By enforcing trust-region chopping as a reaction to convergence issues rather than for every iteration, we obtain a solver more suited for the general case in which convergence failures are not always present. In such cases, oscillation detection is a simple way to significantly reduce the number of iterations.

Numerical experiments

We will validate the trust-region approach on several different models, ranging from simple conceptual models to a full field model. To compare and contrast, we also include a standard sequential method that successively chops the time step until the Newton solver is able to converge. For brevity, we will refer to the former solver as *trust region*, and the latter as *Newton*. All numerical experiments were performed using the open-source MRST software (Bao et al., 2017; Krogstad et al., 2015; SINTEF, 2016).

One-dimensional displacement

We consider a homogeneous $100 \times 1 \times 1$ m channel, in which we inject water at one end, and produce at the other. The channel is tilted by an angle 45° around the y axis to induce gravity effects. The injected water has a viscosity of 1 cP, the oil viscosity is 100 cP, and the relative permeability curves

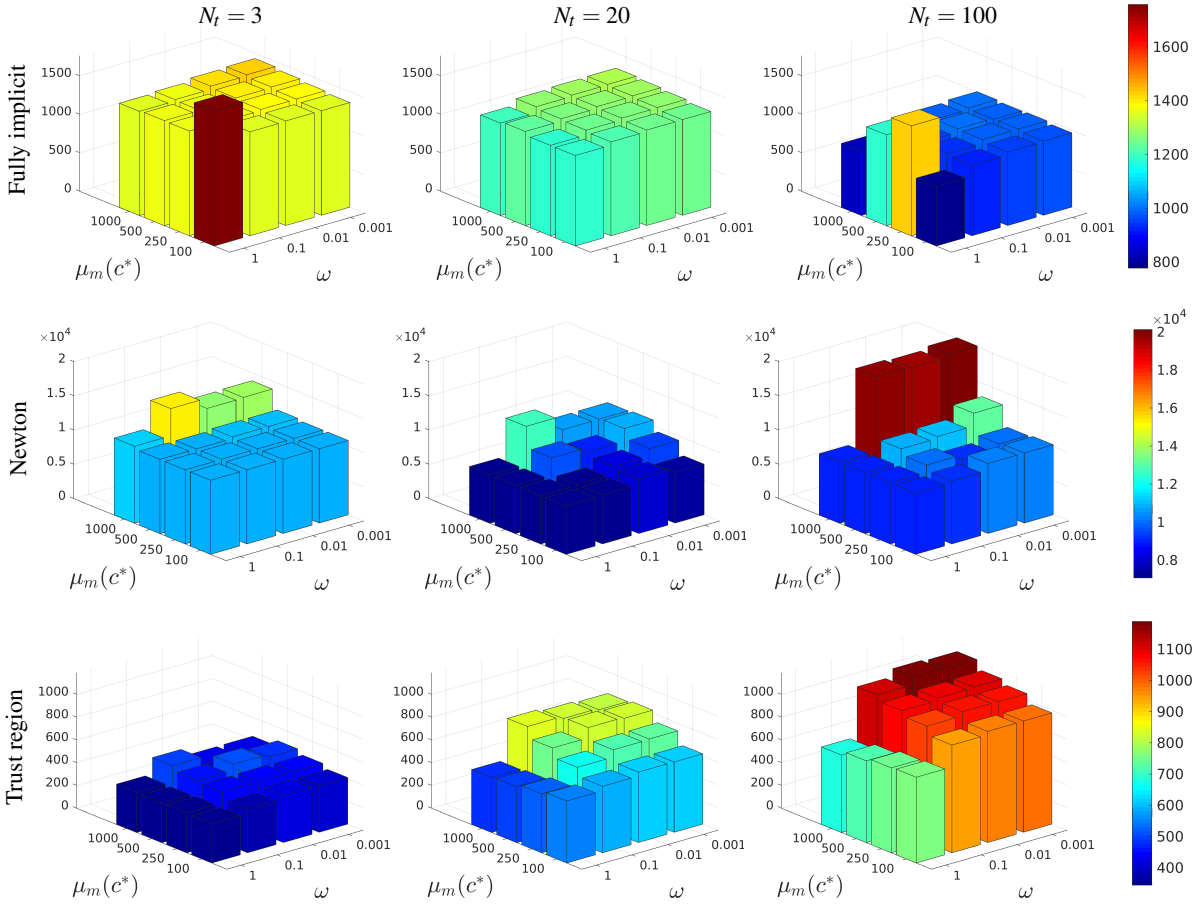


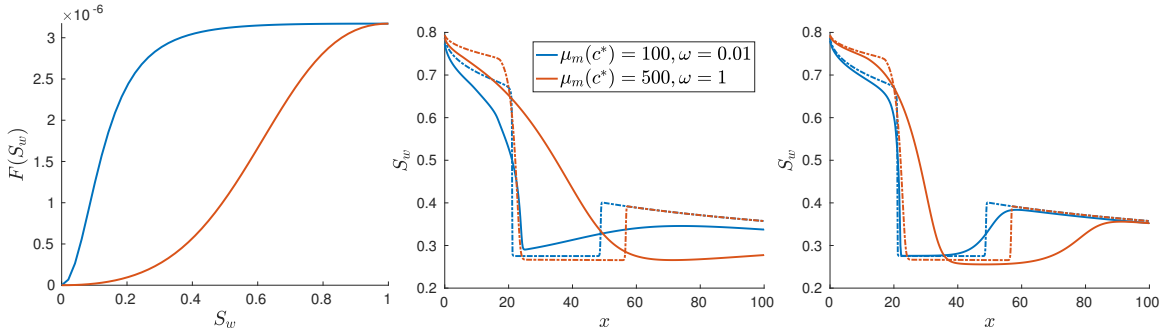
Figure 3: Number of iterations used by a fully implicit simulator and a sequential simulator with Newton or trust-region solver for varying values of $\mu_m(c^*)$ and ω and varying number of time steps N_t. Notice that the color scale and the vertical scale are different for the three solvers.

are quadratic. From 20% to 40% of the injection period, we inject a single polymer slug, while the producer is operated at a fixed bottom-hole pressure. We study how varying values of the maximum viscosity $\mu_m(c^*)$ and the mixing parameter ω affect the number of nonlinear iterations. For comparison, we also report the number of Newton iterations required by a fully implicit simulator which solves both the pressure and saturation equations simultaneously in each time step. Each iteration here involves a larger linear system and is thus more computationally expensive.

The total number of iterations for various combinations of $\mu_m(c^*)$ and ω, and for 3, 20, and 100 time steps, are shown in Figure 3. We observe that the fully implicit solver is less volatile to parameter variations, and the total number of iterations decreases with decreasing time-step length as expected. For the sequential Newton and trust-region methods, on the other hand, the number of iterations increases with the number of time steps and increases significantly towards maximum viscosity and low mixing. In particular, both methods requires fewer iterations for $\omega = 1$, which corresponds to a fully mixed system. This is not surprising, as we see from (5) that in this case, the viscosity multiplier $m(c)$ reduces to unity, effectively removing part of the nonlinearity in the flux functions. Moreover, apart from the fully-mixed case, the effect of maximum viscosity dominates the effect of the mixing parameter. Finally, we observe that the number of iterations used by the Newton method outnumbers the trust-region iterations by more that a factor ten. The reason is that the Newton solver has to repeatedly halve the time step until it is able to find a time step that successfully converges. In effect, the standard Newton method thus wastes a large number of iterations as the iteration process starts over whenever the time step is cut.

While robustness is the main concern of this paper, we are also interested in the accuracy of the simulator.

Figure 4 shows saturation profiles computed on a grid with 500 cells using 20 and 100 time steps for two different polymer parameter combinations. For comparison, reference solutions computed with twice the grid resolution and 10 000 time steps are also included. We clearly see how different polymer setups yield very different flux functions and affect the shock speeds in the saturation profiles. Using 100 time steps gives solutions that are significantly more accurate than using 20 steps. Likewise, the solutions for fully mixed polymers ($\omega = 1$) are less accurate than the solutions with lower mixing. This is due to a so-called self-sharpening effect for the case with $\omega < 1$, giving higher accuracy.



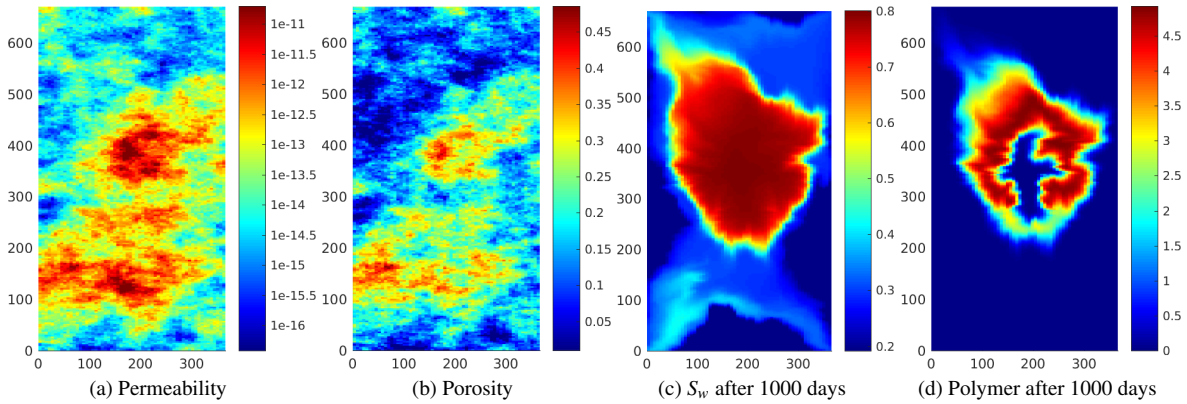
(a) Flux functions for $c = c_{\max}/2$. (b) Solutions using 20 time steps. (c) Solutions using 100 time steps.

Figure 4: Flux functions and solutions for two different polymer setups computed on a grid with 500 cells. Dotted lines are reference solutions computed with 10000 time steps and 1000 cells.

Subset of SPE 10 Model 2

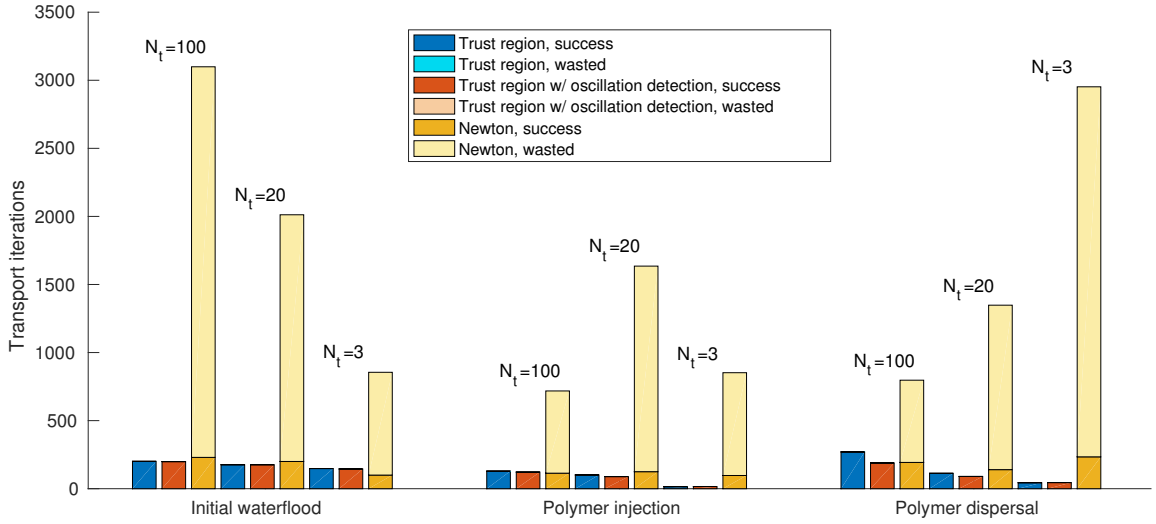
The SPE 10 Model 2 problem (Christie and Blunt, 2001) describes a weakly compressible waterflood problem with an inverted five-spot well pattern over a period of 2000 days. We pick a horizontal layer from the model and inject a single polymer slug after 20% of the injection period and let the producers operate at fixed bottom-hole pressures. The injected volume has been scaled to the fraction of pore-volume in our chosen layer. Conceptually, this example is similar to the one-dimensional displacement discussed above, but the addition of highly heterogeneous permeability and porosity ensures that the problem is more difficult to simulate. Because of the interplay between the global flow field and the local varying properties of individual cells, different regions experience very different nonlinear behavior. Buoyancy effects are not included since the single layer is completely horizontal.

Figure 5 shows permeability and porosity, together with the water saturation and polymer concentration after 1000 days. We simulate the same problem with three different schedules. The first schedule uses one hundred time steps of 20 days each, whereas the second schedule uses twenty steps of 100 days each. The third schedule uses only three time-steps: One for the initial waterflood, another for the polymer injection, and finally one for the period after the polymer slug has been injected. Such a schedule would

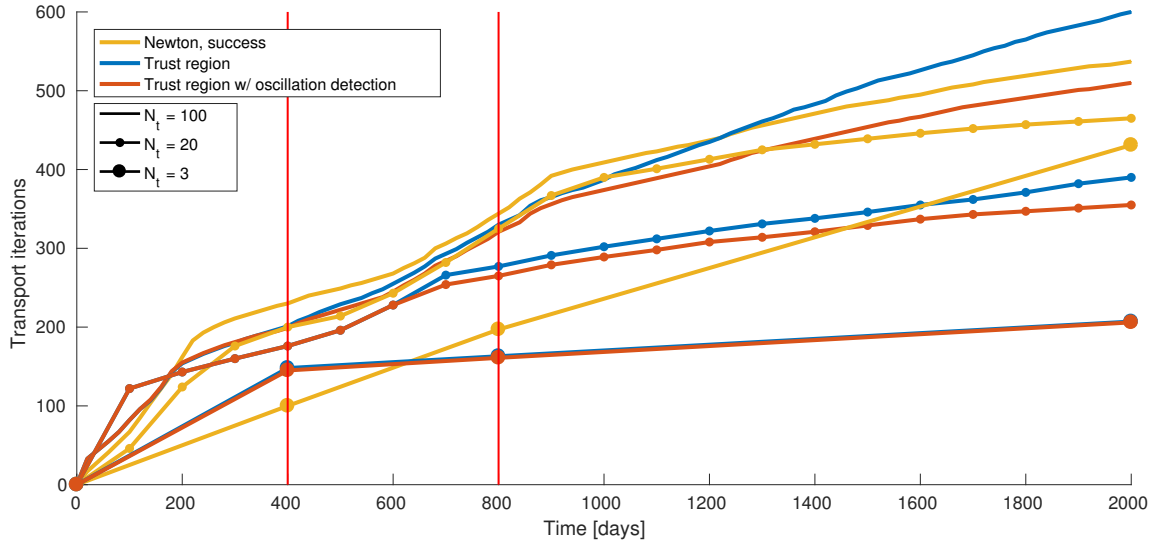


(a) Permeability (b) Porosity (c) S_w after 1000 days (d) Polymer after 1000 days

Figure 5: Petrophysical properties and solution profiles for the SPE 10 subset example.



(a) Total number of iterations per solver for each part of the schedule.

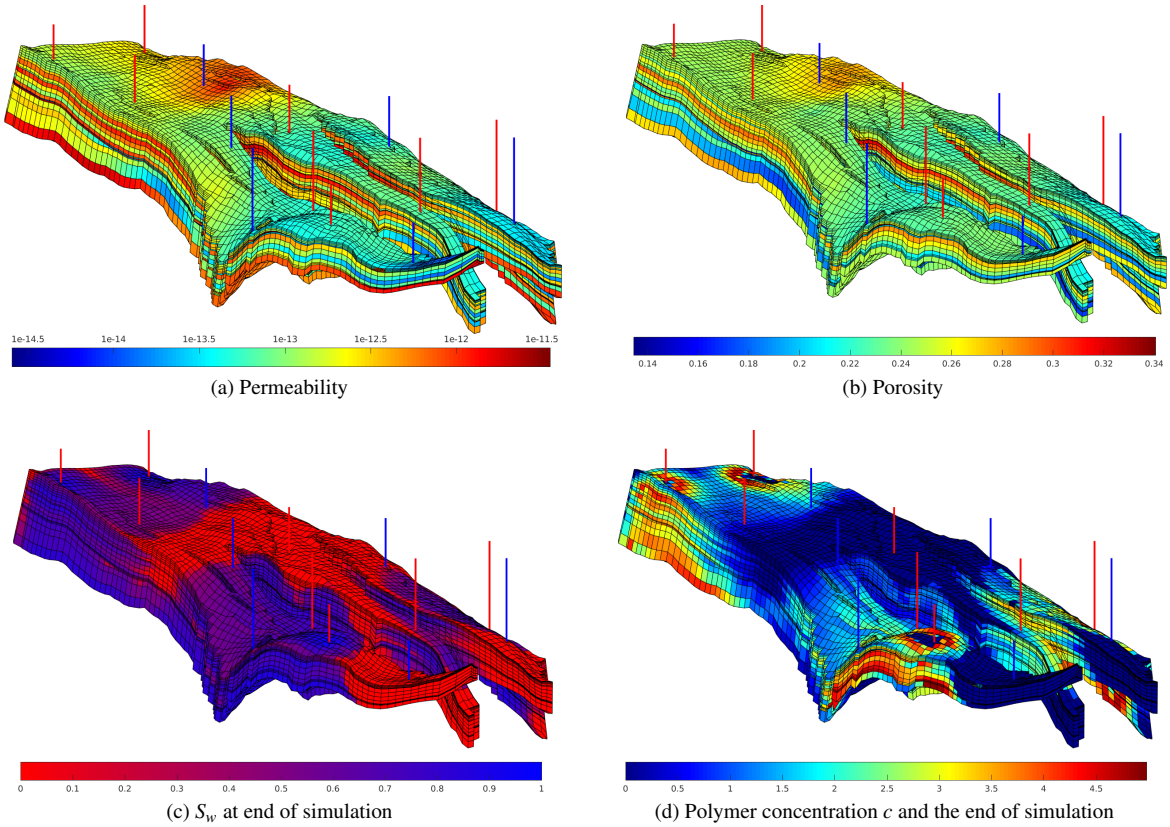


(b) Total number of successful iterations as a function of time for the three different schedules. Vertical red lines indicate polymer injection period.

Figure 6: Overview of iterations used by the transport solvers for the SPE 10 example.

never be used in practice, but is included as an extreme test of robustness. All three schedules are simulated with the Newton and trust-region solvers. Figure 6 reports the resulting number of iterations.

We observe that the trust-region solver does not need to cut time steps. Furthermore, even when the Newton solver has found the largest time steps that do converge after a number of time-step reductions, the trust-region solver still uses less iterations overall. Finally, we note that using oscillation detection to turn on and off the trust-region algorithm has no effect on the number of iterations with 3 time steps, while it reduces the number of iterations by approximately 10 % for 20 time steps, and 15 % for 100 time steps. In a practical simulation, one would typically choose at least 100 transport steps to ensure solution accuracy. In many cases, computational efficiency can be improved by not updating the pressure for every time step, and instead use multiple substeps in the transport solver for each pressure step. This is essentially what is done in streamline simulation. Outer iterations can be added to ensure convergence towards a fully implicit solution for cases with strong coupling between pressure and transport, as discussed e.g., in (Hilden et al., 2016).



(a) Permeability

(b) Porosity

(c) S_w at end of simulation

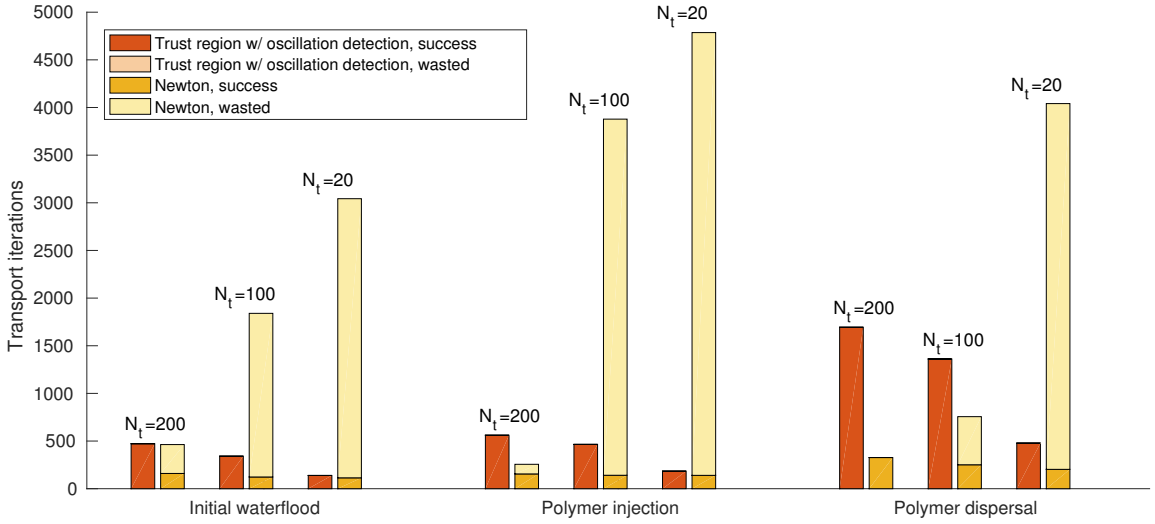
(d) Polymer concentration c and the end of simulation

Figure 7: Petrophysical properties and solution profiles for the Norne example.

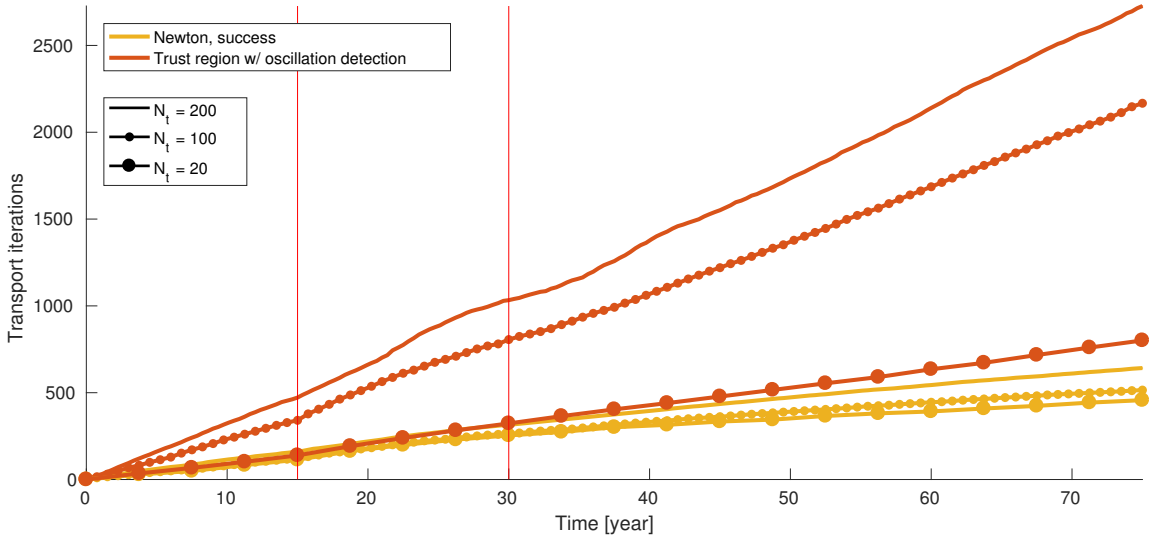
Field model

The last example uses a slightly modified grid model of the Norne oil and gas field (IO Center, NTNU, 2012; The Open Porous Media Initative, 2015) to demonstrate the applicability of the methodology to realistic reservoir geometries and petrophysical properties. After we have removed the three nearly disconnected top layers in the simulation model that mostly contain gas, the grid consists of 37,702 fine cells. Polymer flooding has never been used at Norne, and instead of using the wells from the real simulation model, we set up a completely artificial well pattern consisting of eight injectors and six producers distributed throughout the whole reservoir volume. The wells are vertical and completed in all layers of the model, with a pattern that introduces flow over the entire reservoir so that the evolving displacement fronts pass through as much of the heterogeneity and complex cell geometries in the model as possible. Since some of the wells are completed in regions that are poorly connected with the reservoir, we ended up constructing a very long and contrieved injection horizon of 75 years to avoid unphysically large pressure build up in the wells. Polymer is injected in the period from Year 15 to Year 30 in all injectors. Figure 7 shows reservoir geometry and petrophysical properties along with the water saturation and polymer concentration at the end of the simulation period. The fluid model assumes viscosity of 1 cP for water and 10 cP for the oil phase. Relative permeabilities are Brooks–Corey with exponent 3 and residual saturation of 0.15 for oil and exponent 2 and residual saturation 0.1 for water. The densities are 600 and 1000 kg/m^3, respectively. Both phases are assumed to be incompressible, with weak rock compressibility of 10^{-6} psi^{-1}. The polymer is assumed to be fully mixed with water, giving a water-polymer viscosity of 100 cP at the maximum injected concentration.

We simulate the scenario using three different schedules with 200, 100 and 20 uniform time steps, respectively. Figure 8 reports the total number of iterations for the Newton and the trust-region solvers, whereas Figure 9 shows production curves for the schedules using 100 and 200 time steps. Well curves computed with 100 time steps are very close to those computed with 200 time steps, indicating that both



(a) Total number of iterations per solver for each part of the schedule.

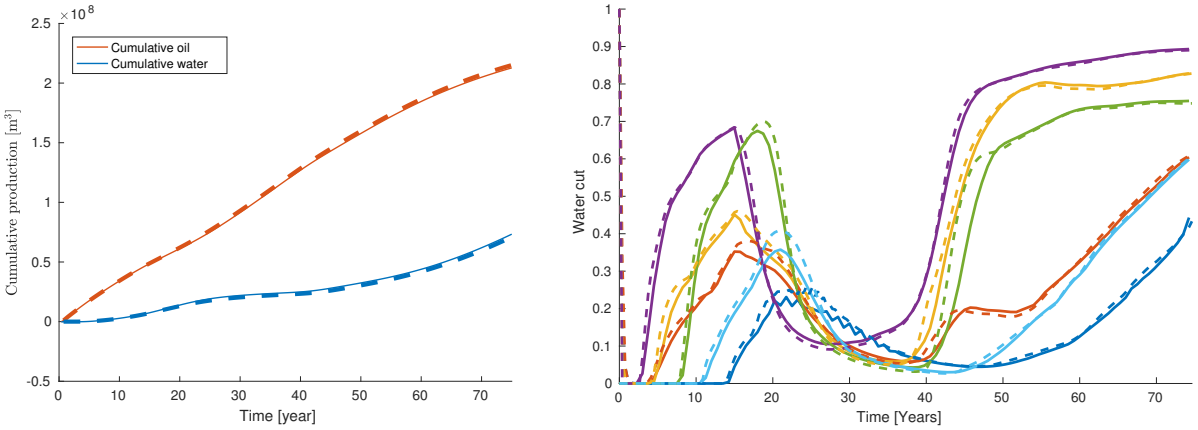


(b) Total number of successful iterations as a function of time for the three different schedules. Vertical red lines indicate polymer injection period.

Figure 8: Overview of iterations used by the transport solvers for the Norne example

these time-step choices can be considered plausible. On the other hand, we emphasize that attempting to use 20 time steps is a difficult test far from the setup that will be encountered in practical computations, and is primarily included to highlight the remarkable robustness of the trust-region method.

With 200 time steps, the standard Newton solver works very well during all three phases of the simulation. The trust-region solver, on the other hand, is too conservative (as illustrated in Figure 2. This is particularly evident during the injection of the polymer slug and in the post-polymer (dispersal) period, where the solver uses almost twice the number of iterations used by the standard Newton solver, wasted iterations included. However, when the number of time steps is reduced to 100 (i.e., decreased by a factor two), we see a dramatic increase in the number of wasted iterations for the standard Newton solver. For the trust-region solver, the average number of iterations per time step increases, but the overall number of iterations is 80 % of the total observed for 200 steps. With 20 steps, the total number of iterations is decreased to approximately 30 %.



(a) Cumulative field oil and water production

(b) Water cut in all producers.

Figure 9: Production curves for the Norne field example. Dashed lines: 200 time steps, solid lines: 100 time steps.

Concluding remarks

We have discussed the trust-region method recently proposed by Møyner (2016) and shown how it can be applied to simulate polymer-flooding scenarios. A number of numerical experiments, three of which are reported herein, demonstrate that the method is robust for almost any time step and for a wide range of polymer parameters. We have compared the new solver to an optimized Newton solver with industry-standard chopping methods. When this solver is working well (i.e., for sufficiently short time steps), it generally requires less nonlinear iterations than the trust-region method. However, the Newton solver is much more sensitive to the choice of time steps, and the number of wasted iterations will often increase dramatically if the time step is chosen too large. This has a strong adverse effect on efficiency and will in many cases also cause the simulator to halt down if the chopping mechanism reduces the time steps too much. This is particularly evident for cases with strong media contrasts and complex reservoir geometries. For the trust-region solver, on the other hand, the number of iterations per time step has only increased moderately with the time-step size in all experiments we have run.

Even though the trust-region method may not always be more efficient than a standard Newton solver, the increased robustness may be beneficial in various applications. One example is the simulation of highly detailed geocellular models with strong media contrasts and complex grid geometry and topology. Here, standard methods often struggle to converge properly because of large variations in CFL number between cells, small inter-cell areas, etc. With the unconditional convergence of the trust region algorithm, we are guaranteed that the solver will converge even for cells with very large CFL numbers, thereby providing the robustness necessary to get a simulation through. Absolute control of the time step length is also important when comparing different polymer models: Whereas a standard method using time-step chopping may end up simulating two different models using potentially very different time-step lengths, the trust region algorithm guarantees that all simulations are performed using the prescribed time-step sizes. Another example is ensemble simulations, for which it is important that the simulator manages to simulate all ensemble members (with comparable time-step sizes) to avoid introducing bias in the ensemble averages. A third example is optimization workflows, where the optimization method may perturb the system outside of the parameter region where the Newton solver works well. The ability to take large time steps is also beneficial in proxy simulations, where one might use a single or a few very large, implicit time steps to obtain representative flux fields to account for e.g. polymer injection effects on the time-of-flight and tracer distributions (Krogstad et al., 2016).

Finally, we have demonstrated how the efficiency of the nonlinear solver can be further improved by turning the trust-region algorithm on and off depending on whether oscillations are present in the Newton updates. This has shown to be particularly beneficial for shorter time steps, when the updates are

relatively small. An obvious extension of our work is to formulate a method that uses a standard Newton method as default, with a trust-region solver as fall-back strategy whenever the convergence of the Newton method deteriorates.

Acknowledgments

The authors were supported by the Research Council of Norway under grant no. 244361.

References

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

Bao, K., Lie, K.A., Møyner, O. and Liu, M. [2017] Fully implicit simulation of polymer flooding with MRST. *Comput. Geosci.* Accepted.

Brenier, Y. and Jaffré, J. [1991] Upstream differencing for multiphase flow in reservoir simulation. *SIAM J. Numer. Anal.*, **28**(3), 685–696.

Christie, M.A. and Blunt, M.J. [2001] Tenth SPE comparative solution project: A comparison of upscaling techniques. *SPE Reservoir Eval. Eng.*, **4**, 308–317.

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

Hilden, S.T., Møyner, O., Lie, K.A. and Bao, K. [2016] Multiscale simulation of polymer flooding with shear effects. *Transp. Porous Media*, **113**(1), 111–135.

Hoteit, H. and Chawathé, A. [2016] Making field-scale chemical enhanced-oil-recovery simulations a practical reality with dynamic gridding. *SPE J.*, **21**(6), 2220–2237.

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

Jenny, P., Tchelepi, H.A. and Lee, S.H. [2009] Unconditionally convergent nonlinear solver for hyperbolic conservation laws with S-shaped flux functions. *J. Comput. Phys.*, **228**(20), 7497–7512.

Krogstad, S., Lie, K.A., Møyner, O., Nilsen, H.M., Raynaud, X. and Skaflestad, B. [2015] MRST-AD – an open-source framework for rapid prototyping and evaluation of reservoir simulation problems. In: *SPE Reservoir Simulation Symposium, 23–25 February, Houston, Texas*.

Krogstad, S., Lie, K.A., Nilsen, H.M., Berg, C.F. and Kippe, V. [2016] Flow diagnostics for optimal polymer injection strategies. In: *ECMOR XV – 15th European Conference on the Mathematics of Oil Recovery, Amsterdam, Netherlands, 29 Aug–1 Sept*.

Li, B. and Tchelepi, H.A. [2014] Unconditionally convergent nonlinear solver for multiphase flow in porous media under viscous force, buoyancy, and capillarity. *Energy Procedia*, **59**, 404–411.

Lie, K.A., Nilsen, H.M., Rasmussen, A.F. and Raynaud, X. [2012] 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*, B25.

Lie, K.A., Nilsen, H.M., Rasmussen, A.F. and Raynaud, X. [2014] Fast simulation of polymer injection in heavy-oil reservoirs based on topological sorting and sequential splitting. *SPE J.*, **19**(6), 991–1004.

Møyner, O. [2016] Nonlinear solver for three-phase transport problems based on approximate trust regions. In: *ECMOR XV – 15th European Conference on the Mathematics of Oil Recovery, Amsterdam, Netherlands, 29 Aug–1 Sept*.

Nasiri, H. and Henriquez, A. [2014] Challenges in implementation of chemical EOR in Norway. FORCE: – EOR competence group: EOR Process Modeling Workshop, 26 May 2014, Stavanger, Norway.

Natvig, J.R. and Lie, K.A. [2008] 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.

SINTEF [2016] The MATLAB Reservoir Simulation Toolbox, 2016b. `http://www.sintef.no/MRST/`.

The Open Porous Media Initative [2015] The Norne dataset. `https://github.com/OPM/opm-data`.

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.

Voskov, D.V. and Tchelepi, H. [2011] Compositional nonlinear solver based on trust regions of the flux function along key tie-lines. In: *SPE Reservoir Simulation Symposium, 21-23 February, The Woodlands, Texas, USA*.

Wang, X. and Tchelepi, H.A. [2013] Trust-region based solver for nonlinear transport in heterogeneous porous media. *J. Comput. Phys.*, **253**, 114–137.