Robust Nonlinear Newton Solver With Adaptive Interface-Localized Trust Regions

Øystein S. Klemetsdal and **Olav Møyner,** Norwegian University of Science and Technology and SINTEF Digital, and **Knut-Andreas Lie,** SINTEF Digital

Summary

The interplay of multiphase-flow effects and pressure/volume/temperature behavior encountered in reservoir simulations often provides strongly coupled nonlinear systems that are challenging to solve numerically. In a sequentially implicit method, many of the essential nonlinearities are associated with the transport equation, and convergence failure for the Newton solver is often 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 updates suggested by the Newton solver if these exceed a predefined limit. Alternatively, one can use trust regions (TRs) to determine safe updates that stay within regions that have the same curvature for numerical flux. This approach has previously been shown to give unconditional convergence for polymer- and waterflooding problems, also when property curves have kinks or near-discontinuous behavior. Although unconditionally convergent, this method tends to be overly restrictive. Herein, we show how the detection of oscillations in the Newton updates can be used to adaptively switch on and off TRs, resulting in a less-restrictive method better suited for realistic reservoir simulations. We demonstrate the performance of the method for a series of challenging test cases ranging from conceptual 2D setups to realistic (and publicly available) geomodels such as the Norne Field and the recent Olympus model from the Integrated Systems Approach for Petroleum Production (ISAPP) optimization challenge.

Introduction

Advances in reservoir simulation continuously challenge the underlying solvers. More-accurate reservoir characterization tends to create grids with high aspect ratios, degenerate cell geometries, small cell interfaces between partially matching neighbors, and orders of magnitude variations in petrophysical properties. The incorporation of physical effects such as enhanced-oil-recovery (EOR) chemistry, temperature-dependent viscosity/density, relative permeability hysteresis, and viscous and gravitational fingering increases the nonlinearities in the flow equations. In addition, realistic reservoir-fluid properties are usually given as tabulated/interpolated data, so that the flow equations are not necessarily pointwise differentiable. Altogether, these and many other factors contribute to create highly nonlinear (and ill-conditioned) discrete systems, which are difficult to solve efficiently using standard Newton or other gradient methods.

Commercial simulators usually rely on a fully implicit discretization of the multiphase-flow equations. This method is unconditionally stable, but inherent iterative linearization gives a large system of mixed elliptic/hyperbolic character that is expensive to solve in a fully coupled manner. To reduce the adverse effects of this mixed character, state-of-the-art constrained-pressure-residual (CPR) methods (Wallis 1983; Gries et al. 2014) use an approximate elliptic equation for the pressure part of the problem as a preconditioner for the full linearized problem.

Sequential-solution approaches reduce computational time by splitting the overall system into a near-elliptic flow equation for pressure and a set of near-hyperbolic equations for the transport of saturations and fluid compositions. One then solves the two subsystems sequentially while keeping certain unknowns fixed (Watts 1986; Trangenstein and Bell 1989). For problems with strong coupling between flow and transport, it is usually necessary to include outer iterations (Jenny et al. 2006; Lu et al. 2007) to ensure that the sequential solution also minimizes the fully implicit residual. The number of outer iterations dictates the overall computational cost, and different methods have recently been proposed to improve the convergence of the outer loop (Jiang and Tchelepi 2018; Moncorgé et al. 2017, 2018). As a viable alternative, one can solve the fully implicit system with the sequential solution as an initial guess and use suitable error indicators to reduce the full system to a small subset of the grid cells (Møyner and Moncorgé 2018).

It is nevertheless important that flow and transport solvers are as efficient and robust as possible. Much research has resulted in highly efficient and scalable pressure solvers that use algebraic multigrid methods (Killough and Wheeler 1987; Trottenberg et al. 2000; Gries et al. 2014) or multiscale methods [see Lie et al. (2017) and references therein] to iteratively and systematically reduce the residual of the flow equation. For transport equations with fixed pressures, the strong hyperbolic character implies localized updates per timestep, but this does not necessarily guarantee fast convergence of the nonlinear solver. Poor convergence can indeed be observed for Newton-type solvers, even for small timesteps and smooth relative permeability curves (Jenny et al. 2009; Møyner 2017). Experience also shows that as geocellular models become increasingly complex and detailed, it becomes more difficult to converge the transport equations than it does the flow equation. It is therefore imperative to increase the robustness and improve the efficiency of implicit transport solvers so that one can either avoid the need to take multiple transport steps per pressure step or do this as efficiently as possible.

A number of methods have been proposed to improve the convergence rates of nonlinear transport solvers, such as use of Appleyard chopping (Schlumberger 2013) to safeguard updates, or more-sophisticated approaches such as localized nonlinear iterations (Younis et al. 2010). Jenny et al. (2009) noted that inflection points in the flux function might send the Newton update to different contraction regions and effectively result in oscillations and convergence issues. One can overcome this using TRs to determine safe saturation updates using inflection points. The original method detected inflection points using closed-form expressions and proved successful for various two-phase-flow scenarios with buoyancy effects (Wang and Tchelepi 2013) and capillary forces (Li and Tchelepi 2014) as well as for compositional flow (Voskov and Tchelepi 2011). Møyner (2017) developed a fully numerical reformulation that detects potential transitions into new contraction regions using an approximate reconstruction of the flux function along the update direction suggested by the Newton solver. The method offers unconditional convergence for multiphase and multicomponent problems with general

Original SPE manuscript received for review 12 April 2018. Revised manuscript received for review 28 November 2018. Paper (SPE 195682) peer approved 12 February 2019.

property curves but is overly restrictive and can result in unnecessarily small updates and wasted computational effort. In this work, we improve the method by introducing adaptivity in the TR solver, switching it on and off depending on observed oscillations in the Newton updates. We demonstrate the performance of the new method on a variety of cases that include realistic geomodels and injection scenarios, as well as conceptual tests constructed especially to challenge the nonlinear solver.

Model Equations

To simplify the description of the new TR method, we only present details for a two-phase oil/water system. Extension to more phases and components follow in the exact same way, with obvious modifications to the discrete transport equations (described later in this paper). The starting point is a standard black-oil model. By introducing a backward Euler temporal discretization, we write the mass-conservation equation for each fluid phase in the semidiscrete residual form,

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

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

where ρ_α denotes density; p_α is pressure; $\vec{v}_\alpha$ is the Darcy velocity; S_α is saturation; μ_α is the viscosity of phase α; ϕ and $\boldsymbol{K}$ are the porosity and permeability of the rock, respectively; z is the vertical coordinate; q_α is the sources and sinks (wells); g denotes gravity acceleration; and $k_{r\alpha}$ models reduced permeability for one phase in the presence of the other. 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 modeled using shrinkage factors, $\rho_\alpha = b_\alpha(p_\alpha)\rho_\alpha^s$, where ρ_α^s is the constant surface density of phase α. A similar relationship, $\phi = b_r\phi_0$, models rock compressibility. We pick oil pressure and water saturation as primary unknowns, which are henceforth denoted as p and S, respectively. We omit capillary pressure to simplify derivations, but these terms are included in our simulator.

Sequential Splitting: Pressure and Transport Equations. To reformulate the system, we first define a pressure equation $R_p = 0$ as a linear combination of the individual residual equations (Eq. 1) weighted by $1/b_\alpha^{n+1}$. This enables us to eliminate the saturations at the end of the timestep using the relation $S_w^{n+1} + S_o^{n+1} = 1$. Likewise, we can reformulate the conservation equations by introducing the total Darcy velocity defined as the sum of the individual phase velocities $\vec{v} = \vec{v}_w + \vec{v}_o$, and using a fractional-flow formulation to obtain new expressions for the fluxes,

$$\vec{v}_\alpha = f_\alpha\left[\vec{v} + \sum_{\beta\neq\alpha}\lambda_\beta(\rho_\alpha - \rho_\beta)g\boldsymbol{K}\nabla z\right],\ \ f_\alpha = \frac{\lambda_\alpha}{\sum_{\beta=o,w}\lambda_\beta}, \quad (3)$$

where $\lambda_\alpha = k_{r\alpha}/\mu_\alpha$ is the mobility of phase α. To obtain a fully discrete model, we introduce a finite-volume discretization with a two-point approximation of the interface fluxes. The transport equations now read

$$R_{\alpha,i} = \left[(\phi b_\alpha S_\alpha)_i^{n+1} - (\phi b_\alpha S_\alpha)_i^{n}\right] + \frac{\Delta t}{V_i}\sum_{j\in\mathcal{N}(i)}|\Gamma_{ij}|(b_\alpha \vec{v}_\alpha\cdot\vec{n})_{ij}^{n+1} - \Delta t(b_\alpha q_\alpha)_i^{n+1} = 0,\ \ \alpha = o, w, \quad (4)$$

where subscript i refers to cell i with bulk volume V_i, subscript ij refers to the interface between cells i and j with area $|\Gamma_{ij}|$, and $\mathcal{N}(i)$ denotes the indices of cells sharing a common interface with cell i. Interface mobilities are evaluated from the upstream direction.

The sequential-solution method starts by solving the pressure equation $R_p = 0$ with fixed saturation to obtain pressure and total velocity. We then solve the transport equation $R_w = 0$ for water to advance saturation a period Δt forward in time, and compute the oil saturation from the closure relation $S_o = 1–S_w$. We repeat this procedure until we reach the desired time horizon. For black-oil-type models (which optionally might contain extra equations to model polymer, alkaline, surfactants, or other chemical species), it is common to solve for the first n–1 phase saturations and all components simultaneously. Another approach is to use a relaxed-volume formulation and instead solve for all n conserved quantities simultaneously, allowing for a deviation from unity in the sum of saturations. This has recently proved to be beneficial for sequential solutions to compositional models (Moncorgé et al. 2018; Møyner and Tchelepi 2018). We can add outer iterations to ensure convergence toward a fully implicit solution for cases with strong coupling between pressure and transport.

TR Algorithm

To simplify notation, we drop the subscript w denoting water and the superscript denoting timestep $n+1$ and introduce a vector notation for the water saturation and the water residuals defined over all N cells in the grid,

$$\boldsymbol{S} = (S_1, \ldots, S_N),\ \ \boldsymbol{R} = (R_1, \ldots, R_N). \quad (5)$$

Solving the transport equations for a timestep of length Δt consists of finding updated saturations $\boldsymbol{S}$ such that the residuals in all cells are zero, $\boldsymbol{R}(\boldsymbol{S}) = \boldsymbol{0}$. We rewrite the residual for the transport equation in cell i as

$$R_i(\boldsymbol{S}) = A_i(S_i) + \sum_{j\in\mathcal{N}(i)} F_{ij}(S_i, S_j) - Q_i = 0, \quad (6)$$

where the accumulation, flux, and sink/source terms read

$$A_i(S_i) = \left[(\phi b S)_i^{n+1} - (\phi b S)_i^{n}\right],$$

$$F_{ij}(S_i, S_j) = \frac{\Delta t}{V_i}|\Gamma_{ij}|\{bf[\vec{v} + \lambda_o(\rho - \rho_o)g\boldsymbol{K}\nabla z]\cdot\vec{n}\}_{ij}^{n+1},$$

$$Q_i = \Delta t(bq)_i^{n+1} = 0.$$

Note that $F_{ij} = -F_{ji}$ for a mass-conservative scheme. To solve the residual equation $\boldsymbol{R}(\boldsymbol{S}) = \boldsymbol{0}$, we use a standard Newton-Raphson method in which the linearized update to $\boldsymbol{S}$ at iteration ℓ is given by

$$\Delta\boldsymbol{S}^{\ell} = -\boldsymbol{J}(\boldsymbol{S}^{\ell})^{-1}\boldsymbol{R}(\boldsymbol{S}^{\ell}),\ J_{i,j} = \partial R_i/\partial S_j. \quad (7)$$

We term a full Newton update as $\boldsymbol{S}^{\ell+1} = \boldsymbol{S}^{\ell} + \Delta\boldsymbol{S}^{\ell}$. It is well-known that Newton's method will experience convergence issues whenever the update passes inflection points in the residual functions R_i. A standard approach is then to successively shorten the timestep until the Newton solver manages to converge; the main drawback is that the nonlinear solver might thus waste many iterations before convergence. Another approach, first introduced by Jenny et al. (2009), is to limit the Newton step using TRs inside which the Newton map is contractive and thus ensured to converge. As an example, consider a scalar equation $h(u) = 0$, for which a TR is an interval (u_l, u_r) on which $h(u)$ is either strictly concave or strictly convex. The idea of the TR method is that the Newton update should not be allowed to pass from one TR and far into another TR within a single iteration, but rather be limited so that the updated value lies immediately inside the boundary of the next TR. In other words, inflection points in the residual function delineate the TRs, and we want to determine the damping factors $\theta_i \in [0, 1]$ so that the updates do not pass far beyond such points:

$$S_i^{\ell+1} = S_i^{\ell} + \theta_i \Delta S_i^{\ell}. \quad (8)$$

Herein, we will follow the approach by Møyner (2017). We consider the interface Γ_{ij} and assume that we have found the Newton increments ΔS_i and ΔS_j for the adjacent cells i and j. These increments define a local update direction,

$$\boldsymbol{d} = \frac{(\Delta S_i, \Delta S_j)}{||(\Delta S_i, \Delta S_j)||}. \quad (9)$$

By considering possible updates in this direction, the process of identifying TRs reduces to a 1D problem in θ.

Inflection points in the residual functions are a primary cause of convergence issues for the Newton solver. For the current two-phase problem (Eq. 6), the accumulation terms A_i are linear, and in many other cases it is simple to show that A_i contain no inflection points. We can thus conclude that convergence issues induced by inflection points mainly stem from the flux function F_{ij}. The directional derivative of this function reads

$$\partial_{\boldsymbol{d}} F_{ij} = \left(\frac{\partial F_{ij}}{\partial S_i}, \frac{\partial F_{ij}}{\partial S_j}\right) \cdot \boldsymbol{d}, \quad (10)$$

with the update direction $\boldsymbol{d}$ defined in Eq. 9. We want to find a TR along the Newton update, and therefore define $\boldsymbol{S}(\theta) = \boldsymbol{S}^{\ell} + \theta\Delta\boldsymbol{S}^{\ell}$ and write $F'_{ij}(\theta) = \partial_{\boldsymbol{d}} F_{ij}[\boldsymbol{S}(\theta)]$. In the same manner, we define the second-order derivative in the direction of the update as $F''_{ij}(\theta) = \partial^2_{\boldsymbol{d}} F_{ij}[\boldsymbol{S}(\theta)]$.

If we have a closed-form expression for F_{ij}, we can use $F''_{ij}(\theta)$ to locate the inflection points of F_{ij} as a function of the damping factor θ. However, second-order derivatives of the flux function are not always well-defined, such as in the case of piecewise linear relative permeabilities. Moreover, they might vanish over large regions of saturations for which the fractional-flow functions are linear or constant. We therefore use a monotone interpolation scheme (Fritsch and Carlson 1980) for $F'_{ij}(\theta)$ to obtain an interpolation $\hat{F}'_{ij}(\theta)$. A TR is then determined by using a modified bisection method in which we evaluate the second-order derivative $\hat{F}''_{ij}(\theta)$ at a small number of sample points (typically three). If $\hat{F}''_{ij}(\theta)$ changes sign between two sample points, we add a sample point at the midpoint of the two. By repeating this procedure a number of times, we obtain estimated inflection points. In all examples herein, we repeat the procedure three times. Kinks stemming from changes in the upwind direction will also result in convergence issues. Therefore, we also identify upwind changes along the update direction, and only allow the upwind direction to change once for each interface during a single iteration. This gives us the information necessary to choose the largest-possible safe update parameter θ_{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 the damping factors θ_i of the TRs equal to the smallest damping of all interfaces,

$$\theta_i = \min_{j,k}\{\theta_{jk}\}. \quad (11)$$

This global chopping strategy offers unconditional convergence, and is the only approach guaranteed not to modify the increment direction $\boldsymbol{d}$. However, it might be far too restrictive. Herein, we follow the approach of Møyner (2017) instead and apply a local chopping procedure, motivated by the observation that transport equations have a strong hyperbolic character. In practice, this means that the largest saturation updates in each nonlinear iteration will (mostly) be localized around displacement fronts, whereas the updates are typically very small when the saturation has small variations in the rest of the domain. As a result, the Newton solver might converge faster in some parts of the domain than in others. To exploit this property, we start by identifying cells that depend strongly on each other using a connectivity matrix $\boldsymbol{C}$, defined as

$$C_{ij} = \begin{cases} 1, & \text{if } |J_{i,j}\Delta S_j| \geq \varepsilon |J_{i,i}\Delta S_i| \\ 0, & \text{otherwise.} \end{cases} \quad (12)$$

That is, the connection across the interface Γ_{ij} is strong if the effect of ΔS_j on the residual relative to the effect of ΔS_i is larger than a given threshold ε. Herein, this threshold has been set to $10^{-6}||\Delta\boldsymbol{S}||_{\infty}$. The matrix $\boldsymbol{C}$ describes a directed graph, and the procedure of finding the global relaxation factors can be summarized with the following steps:

1. Find all cycles (or connected components) in $\boldsymbol{C}$, and combine all the nodes in each cycle into a single supernode. [This decomposition is inexpensive to compute using the Tarjan (1972) algorithm or two depth-first traversals that each visit each node (i.e., cell) in the graph once.]

2. Assign a relaxation factor to each node. For single-cell nodes, this factor equals the smallest relaxation factor of all interfaces with nonzero flux connected to that cell. Nodes consisting of multiple cells are assigned the minimum value of all cells in the cycle.
3. Perform a topological sort of the resulting modified connection matrix with nodes and supernodes. (This is a depth-first traversal of the reduced graph that has N nodes at most.)
4. Traverse the graph and assign to each cell the minimum relaxation factor of itself and all its upstream nodes.

Fig. 1 illustrates the connectivity matrix for 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 because of the density differences. In the original connectivity matrix shown in the lower left in Fig. 1, we have marked cells that belong to the same cycle using distinct colors. These cells show up as blocks on the diagonal in the reordered connectivity matrix shown in the lower right. The localized chopping procedure treats each of these blocks as a single node in the connectivity graph, as described in Step 1.

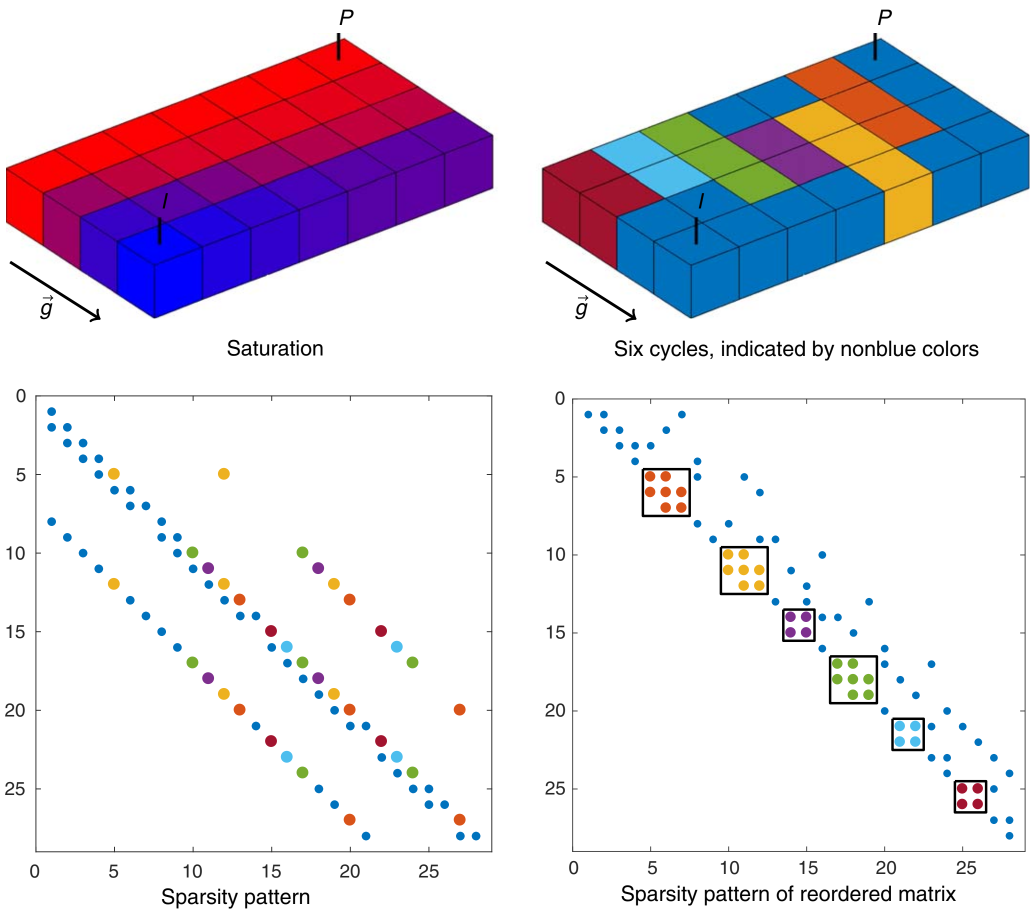

Fig. 1—Cycles and sparsity pattern of the connectivity graph *C* for a problem in which a heavy fluid (blue color) is injected into a lighter fluid (red color). Element C_{ij} is nonzero if an update in cell *j* has a significant effect on the residual in cell *i*. Smaller blue dots correspond to cells that are not part of a cycle, whereas cycle cells are indicated by larger dots, with a unique color for each cycle.

Adaptive TR (ATR) Using Oscillation Detection. Although the TR method ensures that the Newton solver always converges, it might reduce the update even when it is not necessary. This is illustrated in **Fig. 2** for a case where the initial guess and the solution are on the far-opposite sides of an inflection point. In Fig. 2a, the full Newton step passes the solution and the method diverges, whereas the TR solver converges successfully. In Fig. 2b, the solution is shifted slightly to the right so that taking the full Newton step converges much faster than using TRs because the initial (unclipped) linearized update steps over the inflection point and by chance ends up in the same contraction region as the final solution.

Problematic points in the residual function can be identified by oscillations in the Newton updates. To do so, we look at the direction of the Newton increment from Eq. 9 across the interface Γ_{ij}. We say that we have an oscillation over this interface in iteration ℓ if the update vectors $\boldsymbol{d}^{\ell}$ and $\boldsymbol{d}^{\ell-1}$ point to opposite sides of the normal plane defined by $\boldsymbol{d}^{\ell-1}$,

$$\boldsymbol{d}^{\ell} \cdot \boldsymbol{d}^{\ell-1} < 0. \quad (13)$$

For a simple two-phase system, $\boldsymbol{d}$ is a 2D vector, and the oscillation condition (Eq. 13) means that the angle between the two update vectors is greater than $\pi/2$. This information can be used in several ways to determine whether TRs should be invoked. The obvious option would be to invoke TRs over an interface from the first oscillation and keep it on for this interface throughout the timestep.

In practice, TRs are only necessary for a few iterations, and we might want to turn them off again to speed up the solution process. In the worst case, we will have to pass *all* problematic points in the interface flux function before the solver converges, and we therefore suggest a simple adaptive approach: Denoting the number of observed oscillations over an interface by n_{osc}, we invoke TRs if any of the last $n_{\text{osc}} + 1$ iterations (including the current) resulted in an oscillation. The observed oscillations n_{osc} will then be a lower bound on the number of problematic points passed by the solver, and the resulting solver will be better suited for the general case. It will be close to the standard Newton method when it performs well and will approach the static TR solver for particularly challenging residual functions.

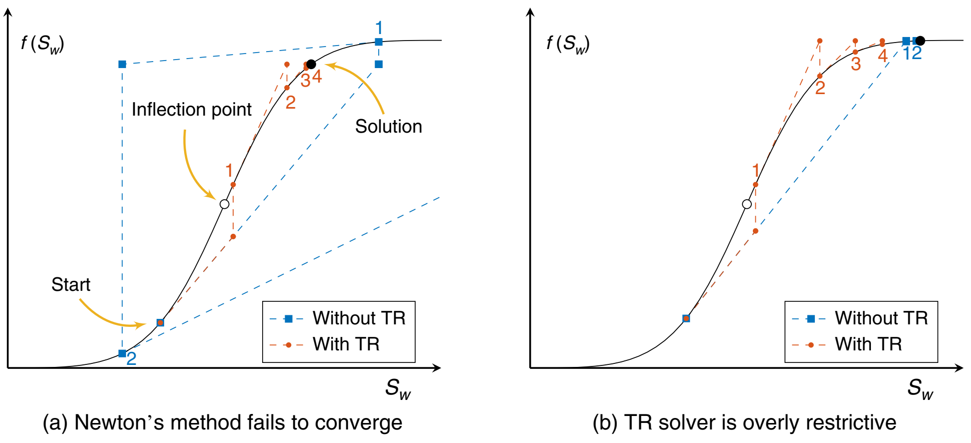

Fig. 2—Illustration of how Newton's method and the TR solver may choose very different iteration paths when applied to the same problem. (a) Newton's method passes the inflection point and fails to converge, whereas the TR solver converges successfully. (b) Both methods converge, but the TR solver is overly restrictive.

The iterations might still fail to converge for very peculiar flux functions because the Newton updates can follow a closed path around the solution as long as $\boldsymbol{d}^\ell \cdot \boldsymbol{d}^{\ell-1}$ is always positive. We can overcome this by invoking regular TRs for all interfaces if the solver has not converged after a predetermined maximum number $n_{\max}$ of iterations, or we can check the oscillation condition (Eq. 13) for the current update direction against the update directions for several iterations back in time. That is, we invoke TRs if

$$\boldsymbol{d}^\ell \cdot \boldsymbol{d}^{\ell-k} < 0 \text{ for any } k = 1, \ldots, n_{\text{chk}}. \quad (14)$$

We set $n_{\text{chk}} = 1$ initially and increase the parameter each time the solver has performed a multiple of $n_{\max}$ iterations without converging. Algorithm 1 summarizes the complete procedure.

Algorithm 1—Adaptive Oscillation Detection Over an Interface

```
1. $\ell \leftarrow 1$                                            ▷ Iteration counter
2. $n_{\text{chk}} \leftarrow 1$                                  ▷ Number of update directions back in time we compare with current
3. $n_{\text{osc}} \leftarrow 0$                                  ▷ Number of observed oscillations over the interface
4. $n_{\text{TR}} \leftarrow 0$                                   ▷ TR invoked for this interface if $n_{\text{TR}} > 0$
5. while Solver has not converged do
6.     if $\boldsymbol{d}^\ell \cdot \boldsymbol{d}^{\ell-k} < 0$ for any $k = 1, \ldots, n_{\text{chk}}$ then   ▷ Oscillation condition
7.         $n_{\text{osc}} \leftarrow n_{\text{osc}} + 1$         ▷ Update number of observed oscillations
8.         $n_{\text{TR}} \leftarrow \max(n_{\text{osc}}, n_{\text{TR}}) + 1$   ▷ Invoke TR for the next $n_{\text{osc}} + 1$ iterations
9.     else
10.        $n_{\text{TR}} \leftarrow \max(n_{\text{TR}} - 1, 0)$  ▷ No oscillation, reduce $n_{\text{TR}}$
11.    end if
12.    if $n_{\text{TR}} > 0$ then
13.        Invoke TR                                             ▷ TR is invoked if any of the last $n_{\text{osc}} + 1$ updates had oscillations
14.    else if mod $(\ell, n_{\max}) = 0$ then
15.        $n_{\text{chk}} \leftarrow 2n_{\text{chk}}$            ▷ We seem to have convergence issues, so we double $n_{\text{chk}}$
16.    end if
17.    $\ell \leftarrow \ell + 1$
18. end while
```

Fig. 3 shows a schematic example of how the ATR solver works. With the default setting (shown in the lower gray box), TRs are not active initially and the first step thus jumps across several inflection points. The second Newton iteration results in an oscillation, and we invoke TRs. For the third iteration, we check Iterations 2 and 3 for oscillations. Because Iteration 2 had an oscillation, we invoke TRs for Iteration 3 as well. Neither Iteration 3 nor 4 resulted in oscillations, and hence no TR check is used in Iteration 4. This takes us past the solution, and Iteration 5 gives a new oscillation. From this iteration, we check the last three iterations for oscillations, and the method successfully converges after seven iterations.

The example also illustrates that more-efficient strategies can be formulated if we know the location of the problematic points or expect that the solver will encounter problems in the first iterations and decide to use the adaptive algorithm in a more-defensive manner. If TRs were turned on for the first step (e.g., by setting $n_{\text{TR}} = 1$ initially), the iteration will converge in two steps, as shown in the upper gray box. By adjusting the initial setting of the n_{TR} parameter, we can determine how restrictive the algorithm is; the original TR method is obtained by setting n_{TR} to a large positive number. In general, the adaptive algorithm is aimed at handling tabulated property curves typically encountered in real models, for which one will not generally know the location of problematic points a priori. In our experience, the numerical TR algorithm from Møyner (2017) is overly cautious and has a tendency to identify points along the Newton path as problematic even if they are not. Likewise, the damping factor θ_i is determined as the minimum over all upstream cells, which might be overly restrictive for some of the interfaces. The adaptive approach is a way to improve performance and present an algorithm that is flexible, robust, and reasonably efficient.

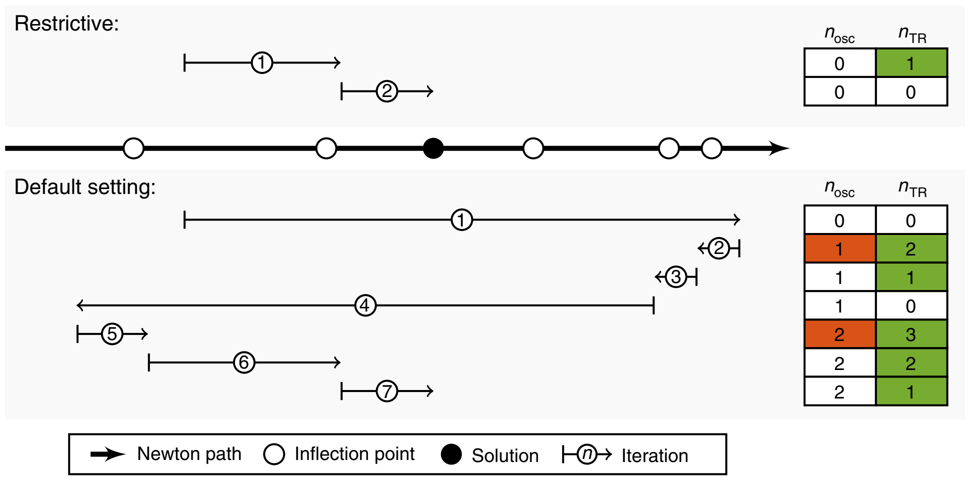

Fig. 3—Schematic example of the ATR solver using oscillation detection. The gray boxes show possible sequences of Newton updates over an interface obtained with two different algorithmic settings for a flux function with five inflection points. The table on the right-hand side shows the number of oscillations (n_{osc}) and the number of iterations (including the current) for which TR is active over the interface (n_{TR}).

For simulations using (very) large timesteps, the initial updates will be large and are likely to pass problematic regions in the flux function. In such cases, it might be safer to start with TRs on and rather disable them if no problems are detected. This is achieved when the initial value of n_{TR} is set to a positive number. With more-modest timesteps, the Newton solver will usually experience convergence issues only for a few timesteps, and it is better to start with $n_{TR} = 0$ so that TRs are enforced as a reaction to convergence issues rather than as a default for every iteration. This way, we obtain a solver that is equally robust, but more efficient in the general case. We note that for all examples we have run, it has been sufficient to compare the current update direction with the previous one, so that n_{chk} never gets a value greater than unity for any interface in any iteration. That is, the fallback strategy in Line 15 in Algorithm 1 has never been reached. One can also think that for very long timesteps, it might be beneficial to invoke TRs for more than $n_{osc} + 1$ iterations when n_{osc} is small because it is likely that TRs are needed for almost all iterations and all interfaces. However, in all the examples we have run with very long timesteps, we have found that it is sufficient to always invoke TRs for $n_{osc} + 1$ iterations to closely match the number of nonlinear transport iterations used by the static TR method. This will be illustrated in Polymer Example 1 in the next section.

Numerical Experiments

This section validates the static TR algorithm and the ATR algorithm on several different cases, ranging from simple conceptual models to field models. The test cases feature two-phase-flow and compressible three-phase-flow physics, as well as a standard Todd and Longstaff (1972) type of model for polymer flooding. Detailed descriptions of the corresponding sequential-solution procedures are provided by Hilden et al. (2016) and Møyner and Lie (2016).

The TR method of Møyner (2017) and our new ATR method were both implemented in the open-source MATLAB® Reservoir Simulation Toolbox (MRST) (Lie 2019; Krogstad et al. 2015). Although not a commercial simulator, MRST implements many of the same models, discretization methods, and solution algorithms seen in commercial reservoir simulators. Using MATLAB introduces certain computational bottlenecks not seen in simulators written in a compiled language, but MRST has been extensively validated and benchmarked against leading commercial simulators and has been shown to converge to the correct solution at expected rates on a wide variety of test cases, from simple benchmarks to field-scale asset models. We thus believe that MRST constitutes a reliable and representative test bench for new nonlinear-solution algorithms.

As a baseline for the comparisons, we consider a plain Newton solver and an improved Newton solver with line search along the Newton path to reduce the residual when experiencing convergence problems; we refer to these as Newton and LS, respectively. Neither of these solvers is guaranteed to converge for any timestep, and to remedy this we incorporate parts of the well-known modified Appleyard-chop algorithm by ensuring that saturations are between zero and one and that changes in saturations are kept to less than 0.2. However, to keep the comparison as straightforward as possible, we do not cut back the updates when saturations move from an immobile to a mobile state, or vice versa. In all dynamic simulations, except in the first example, the Newton solvers are also set to halve the transport step if the iteration does not converge within a prescribed number of iterations.

The solvers use the same convergence criteria in all experiments, and it is thus natural to use the number of nonlinear iterations required for convergence as an indicator of computational performance. Using modern linear solvers, the cost of solving the linear system is nearly $N^{1.2}$ for N unknowns. In our 3D experiments (Norne and Olympus), we use the C++ AMGCL solver library to solve the linear systems regardless of the nonlinear-solver choice (Demidov and Rossi 2017; Demidov 2018). One can thus argue that the computational cost is comparable with the cost of residual evaluations and that the number of residual evaluations hence can be used as a measure of computational performance.

Two-Cell Problems for Different Fluid Models. We start by comparing the solution paths taken by the Newton solver, Newton with chopping depending on the saturation increment, LS, the original numerical TR method, and the ATR with initial parameters $n_{TR} = 0$ and $n_{TR} = 3$ on a simple two-cell problem with injection in the left cell and production in the right cell. We mimic the process of water displacing oil by injecting a large volume of water during a single timestep for various two-phase-fluid models. After the timestep, both cells will be at irreducible oil saturation. In this particular example, we limit saturation updates to 0.1. To contrast the Newton and the LS solvers, we allow infinitely large saturation updates in the latter and use line search for each iteration.

The first fluid model samples relative permeabilities from the SPE1 benchmark (Odeh 1981), but sets k_{rw} equal to k_{ro} to make water mobile. **Fig. 4** shows solution paths taken by the six solvers, each with four different initial guesses, plotted on the residual surface. The solvers generally follow quite different paths. For the Newton and the LS methods, some solution paths are not shown because they did not converge within 100 iterations. **Fig. 5** reports the number of iterations, limited upward by a maximum of 25 iterations, for 2,500 different initial guesses, sampled at the midpoint of each cell in a 50×50 mesh that covers the unit square. As expected, the Newton method only converges in a small subset of the unit square. Introducing line search doubles the size of the convergent subset, and by chopping saturation increments, we obtain convergence in the whole domain. Compared with the TR solvers, we see that TR is less efficient than Newton with chopping, and ATR with $n_{\mathrm{TR}} = 0$ (the default setting) converges rapidly in large, contiguous patches and is overall the most efficient. In contrast, ATR with $n_{\mathrm{TR}} = 3$ has many of the same characteristics as TR but is more efficient in many points with low S_L or S_R values.

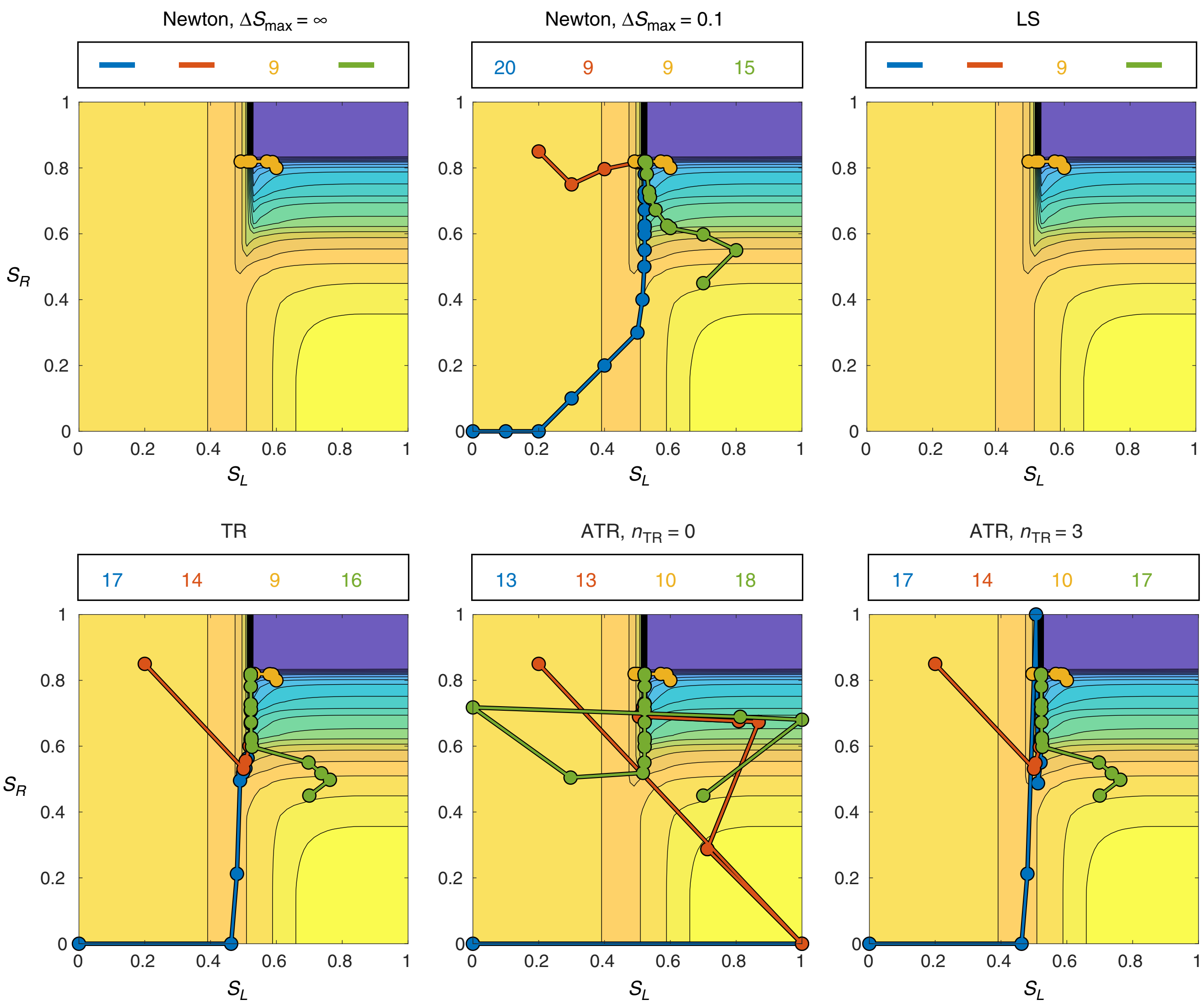

Fig. 4—Newton paths taken by six different nonlinear solvers from four different initial guesses for a two-cell problem with the SPE1 fluid. Solution paths are not shown if the solver did not converge within 100 steps.

The second model describes a two-phase, three-component system with polymer that mixes with water according to the Todd and Longstaff (1972) model for miscible flooding. The relative permeabilities are Brooks-Corey with an exponent of 3 and residual saturation of 0.15 for oil, and exponent of 2 and residual saturation of 0.1 for water. We inject a polymer slug together with water and set the water saturation and polymer concentration in the right cell equal to the injected saturation and concentration. Because the water/polymer mixture will eventually fill both cells, we can plot the residual in the left cell as a function of water saturation and polymer concentration. **Fig. 6** shows the residual and solution paths for four different initial guesses. The Newton and LS solvers do not limit the concentration update. We clearly see that the introduction of polymer makes the transport problem highly nonlinear. Although the unmodified Newton solver only converges for one of the four initial guesses, the Newton solver with the maximum saturation update converges for all starting points. The solution paths are similar for LS and ATR with $n_{\mathrm{TR}} = 0$, as are most of the paths taken by TR and ATR with $n_{\mathrm{TR}} = 3$. Notice, however, that ATR avoids the unnecessary use of TRs by switching them off after Iteration 2 for the path starting at the origin, and as a result converges in nine iterations instead of 16.

Fig. 7 reports iterations for a 50×50 mesh of different initial guesses for S_w and $\hat{c}$. The unlimited Newton solver only converges in a small region where S_w is close to the solution. Likewise, because the Newton solver with $\Delta S_{\max} = 0.1$ does not limit the concentration update, the iteration surface is made up of vertical patches of width $\Delta S_{\max}$. We also see that LS and ATR with $n_{\mathrm{TR}} = 0$ use approximately the same number of iterations as the unlimited Newton solver within its convergence region. Overall, ATR with $n_{\mathrm{TR}} = 3$ is the most efficient solver in this example, but ATR with $n_{\mathrm{TR}} = 0$ is more efficient inside the region of mobile water.

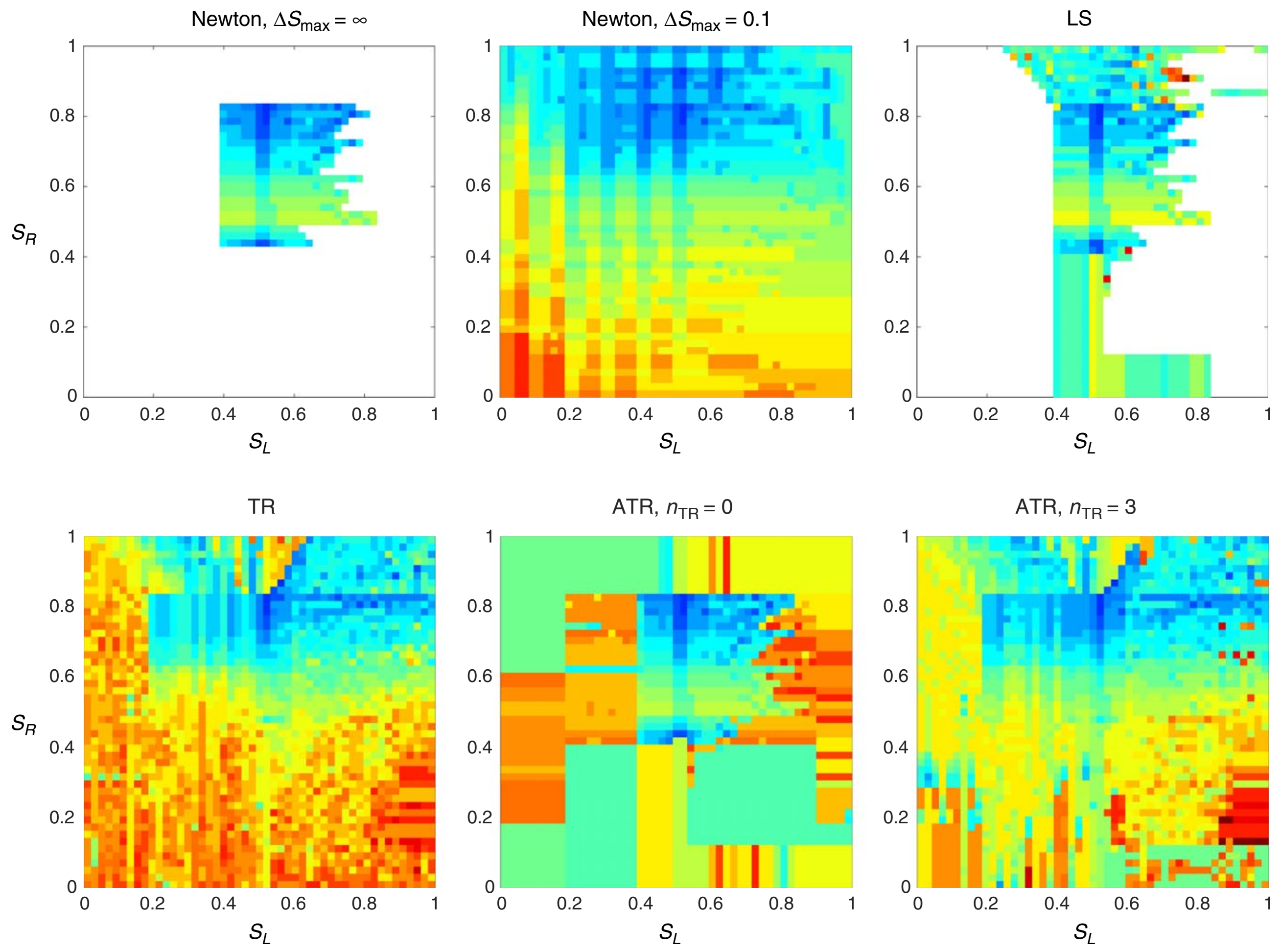

Fig. 5—Number of iterations required to converge on a 50 × 50 mesh of different initial guesses for the two-cell problem with the SPE1 fluid. White color indicates that the solver did not converge within 25 iterations.

The main purpose of this example was to illustrate that nonlinear solvers might take quite different solution paths even in simple cases. On the basis of these few samples, we cannot draw a firm conclusion, but the results indicate that the ATR solver will also be efficient in more-complex multidimensional scenarios where the saturations and concentrations depend on values in upstream cells.

Quarter-Five-Spot Pattern. The next example is a quarter-five-spot pattern posed on a quadratic domain of $2000 \times 2000\,\mathrm{m}^2$ with homogeneous permeability and porosity. The rock is completely filled with oil, and a total of 0.2 pore volumes (PV) of water are injected at a constant rate over a period of 2 years. The producer operates at a fixed bottomhole pressure (BHP) of 50 bar. The fluids are incompressible and follow the first model from the previous example. We use 24 evenly spaced timesteps of approximately 30 days to simulate 2 years of production.

Fig. 8 reports the cumulative number of iterations used by the Newton, LS, TR, and ATR solvers for all transport steps. The TR solver uses the fewest iterations in the first step, during which the first water-displacement fluid enters the reservoir. The Newton solver then uses fewer iterations than the other three in the second and third timesteps, but in Timesteps 4, 6, and 14, it does not converge within the maximum allowed number of iterations. As a result, the timestep is halved until the solver converges, giving several wasted iterations for which partial results are discarded; these are reflected as jumps in the cumulative iteration count. The total number of wasted iterations obviously depends on the prescribed maximum iterations performed before chopping the timestep: If set too high, a large number of iterations are wasted while the solver oscillates between two values, and if set too low, timesteps might be cut unnecessarily. In all examples presented herein, we set the upper iteration limit at 25, which we believe is a reasonable compromise. The reader should keep this in mind when studying the examples herein. Fig. 8 also reports the number of successful iterations performed by the Newton solver. Iterations are termed successful if they were part of a timestep that did not need to be reduced to achieve convergence. These steps represent the idealized case in which we know a priori how long we can choose the timesteps and still obtain convergence. Using LS avoids timestep chops, but requires more iterations than the successful Newton steps.

Fig. 9 shows the water saturation in four selected timesteps, with the interfaces on which TR/ATR are active indicated in magenta. The ATR algorithm reduces the number of active interfaces to as little as 10 to 20% percent of the number used by TR. The effect of this is twofold. First, the algorithm is less restrictive in the sense that it allows for larger saturation updates on many of the interfaces, resulting in faster convergence. Second, it reduces the computational cost because TRs are only used on a very small subset of the grid interfaces, requiring fewer evaluations of the flux function. Note that TR and ATR have no problem converging for the problematic timesteps, and as a result use much fewer iterations overall than standard Newton chopping. In particular, using TR and ATR reduces the total number of nonlinear iterations by 18 and 53%, respectively. In total, ATR uses fewer iterations than the number of iterations over the successful Newton steps. ATR uses fewer or equally many iterations for many of the timesteps that were not chopped. This is an important property because it means that ATR will not use significantly more iterations than a standard Newton solver when this solver performs optimally.

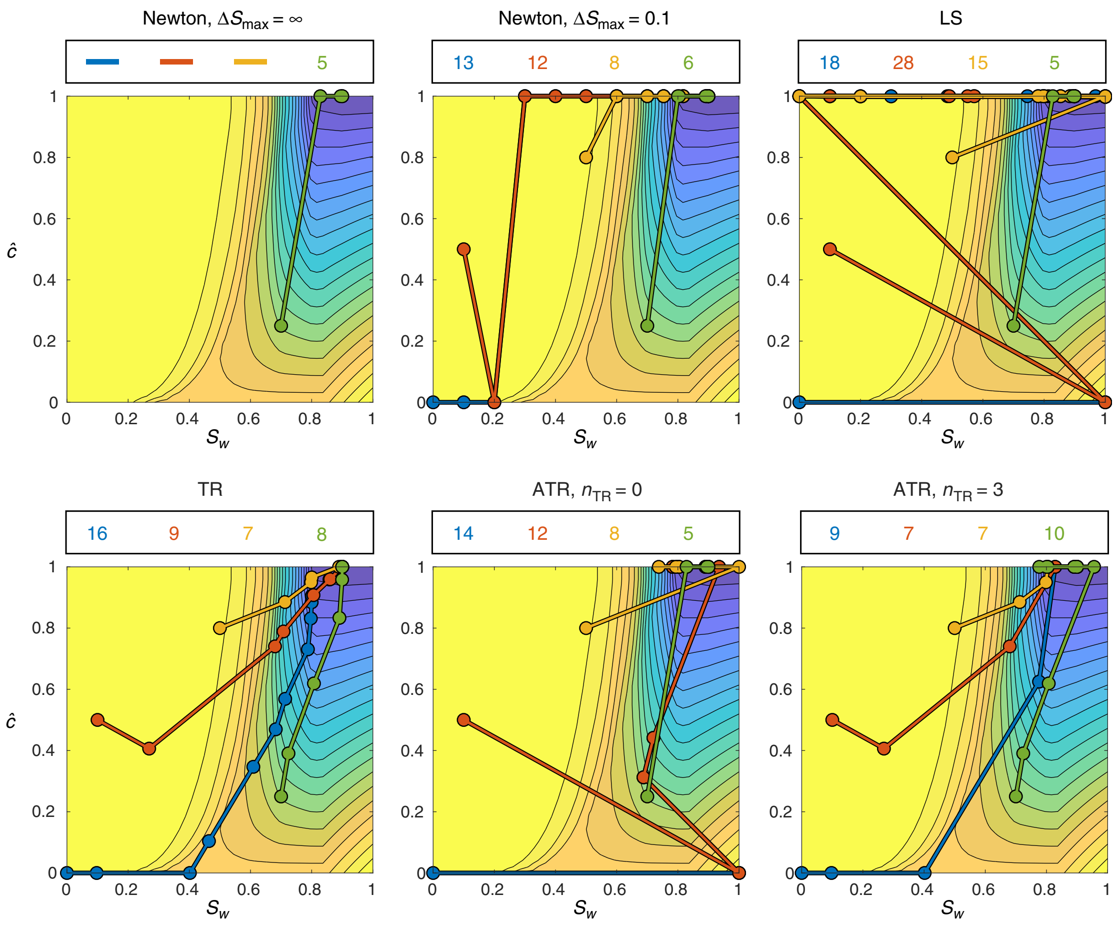

Fig. 6—Newton paths taken by six different nonlinear solvers from four different initial guesses for a two-cell problem with the two-phase three-component polymer model. Solution paths are not shown if the solver did not converge within 100 steps.

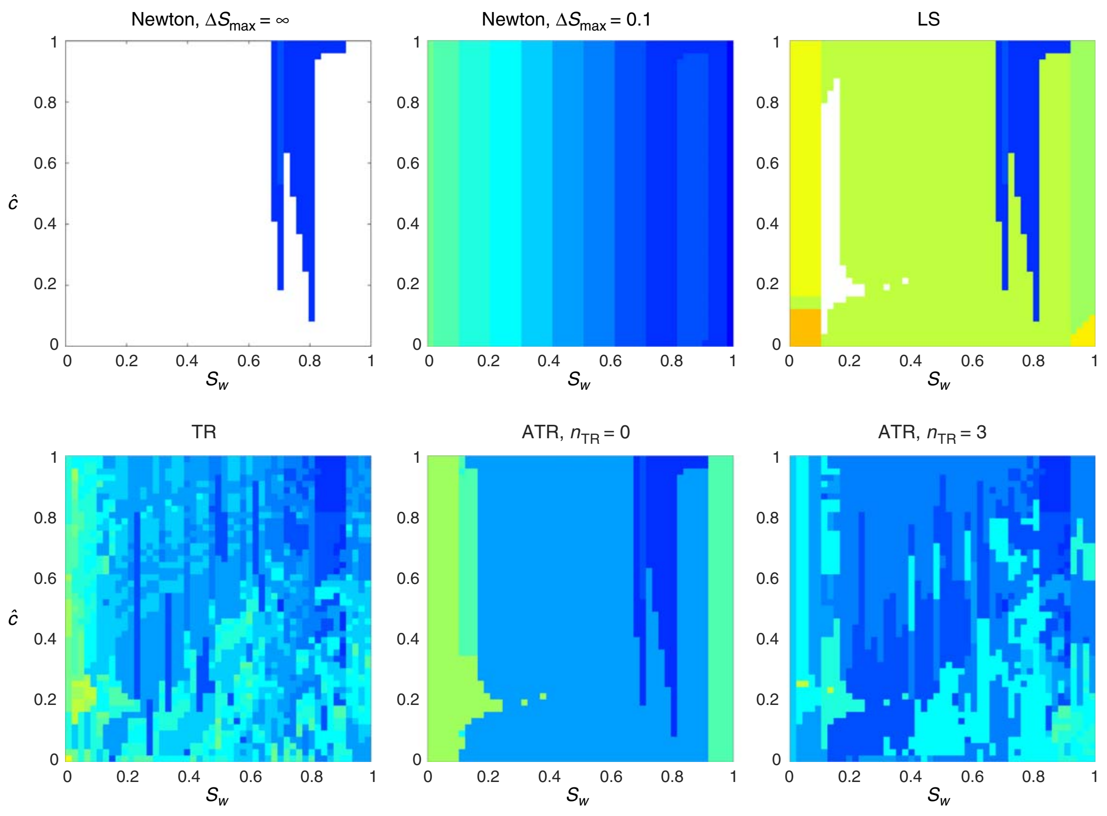

Fig. 7—Number of iterations required to converge on a 50 × 50 mesh of different initial guesses for the two-cell problem with the two-phase three-component polymer model. White color indicates that the solver did not converge within 25 iterations.

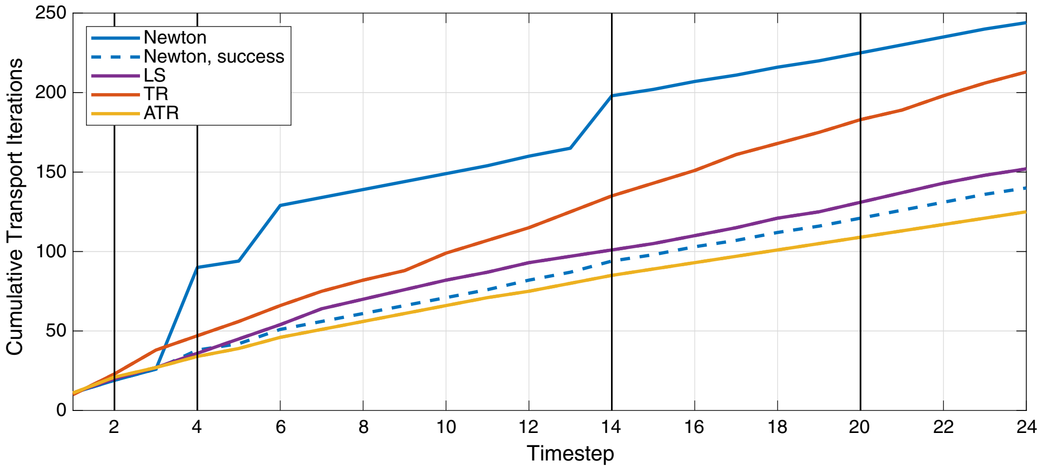

Fig. 8—Nonlinear transport iterations per timestep used in the quarter-five-spot example. Dots indicate iterations per timestep (left axis), and lines indicate cumulative iterations (right axis). Black vertical lines correspond to the timesteps where water saturation and active interfaces are reported in Fig. 9.

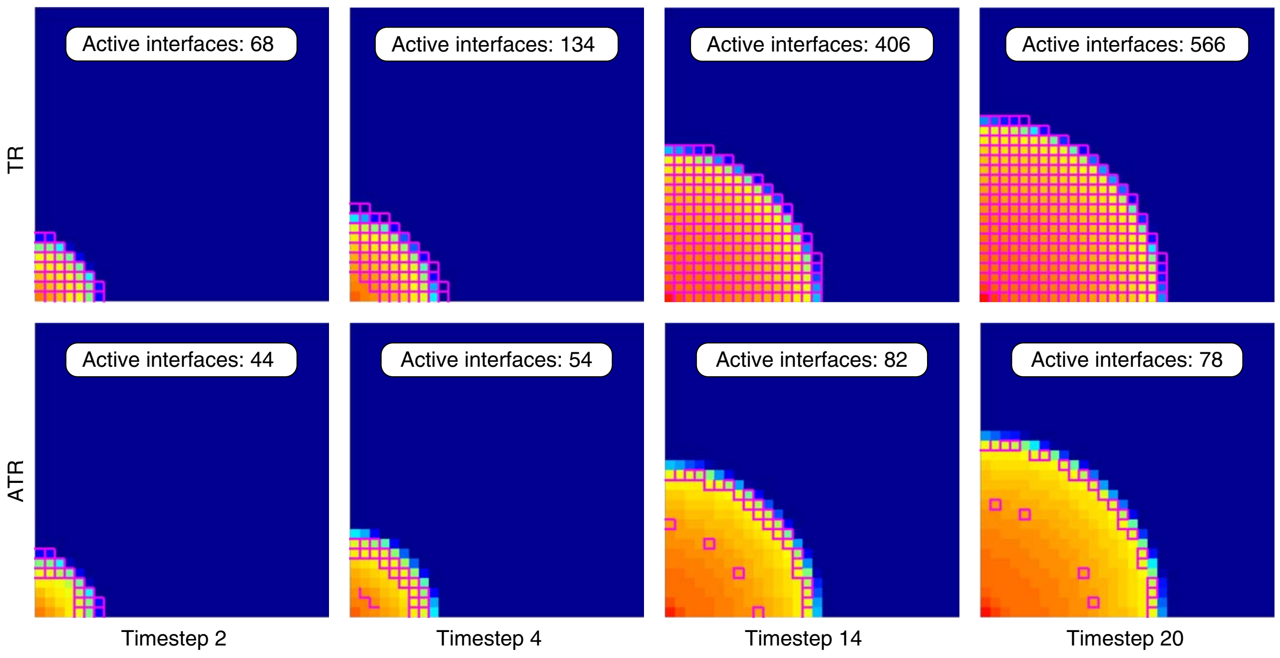

Fig. 9—Water saturation S_w at selected timesteps in the quarter-five-spot example. The interfaces indicated in magenta are the maximum number of active interfaces for the two TR solvers throughout the timestep.

Layered Permeability. Our next example is a conceptual test aimed to challenge the nonlinear solver. The problem is posed on a $2000 \times 50\,\text{m}^2$ vertical cross section in which the permeability is made up of a highly heterogeneous repeating pattern consisting of three rows of cells with permeability of 1,000, 100, and 10 md, respectively. The in-place fluid has a density of $800\,\text{kg/m}^3$, and we inject a total of 0.2 PV of a fluid with density of $1000\,\text{kg/m}^3$ through the upper half of the leftmost boundary and 0.2 PV of a light fluid with density $100\,\text{kg/m}^3$ through the lower half over a period of 2 years. The fluid phases are incompressible and described by a black-oil model with quadratic relative permeabilities. A producer at the rightmost boundary operates at constant BHP of 50 bar. This will induce a complex flow pattern with a combination of viscous fingering in the lateral direction because of the permeability anisotropy and rapid gravity segregation in the vertical direction because of high density differences. Such a problem is very challenging for the Newton solver, which we allow to perform 50 iterations before chopping. **Fig. 10** shows the saturation of the light fluid at selected timesteps.

To investigate the robustness and applicability of the TR solvers, we use four simulations with timesteps of 1/3, 1/6, 1/12, and 1/24 of a year, respectively. **Fig. 11** reports the number of nonlinear transport iterations used by the Newton, LS, TR, and ATR solvers. The Newton solver struggles significantly for the two longest timesteps and wastes a large number of iterations. The TR and ATR solvers, on the other hand, waste no iterations. We also observe that using adaptivity improves the TR solver significantly and that the effect is better for shorter timesteps, when TR is only needed for a few iterations. Indeed, using ATR gives significantly fewer active interfaces in Fig. 10. Notice also that ATR outperforms the Newton solver in terms of iterations, even with timesteps of 15 days. The LS solver performs comparably or slightly better than ATR, except for the setup using the longest timesteps, where it has to halve the timestep to converge.

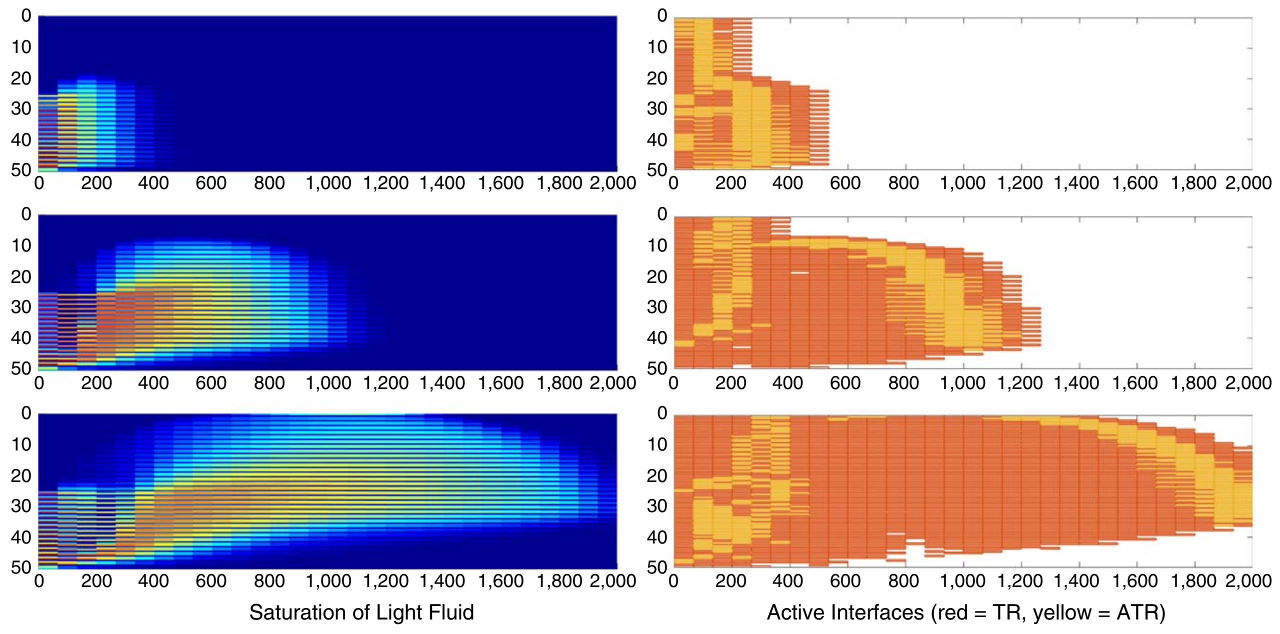

Fig. 10—Saturation of the light fluid (left), along with active interfaces computed using timesteps of 15 days for the two TR solvers at selected timesteps (right) for the layered-permeability example.

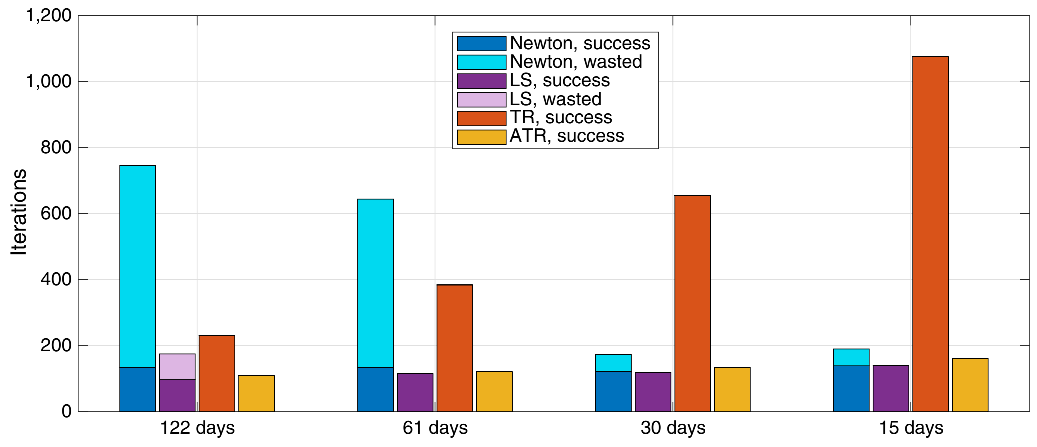

Fig. 11—Iterations for the different timesteps and solvers in the layered-permeability example.

Polymer Example 1: Subset of SPE 10 Model 2. As an example of an EOR process, we consider polymer flooding described by a Todd and Longstaff (1972) model for miscible flow as specified in the ECLIPSE commercial simulator (Schlumberger 2013). We pick a horizontal layer from SPE 10 Model 2 (Christie and Blunt 2001), which describes a weakly compressible waterflood problem, and inject 1 PV of water over a period of 2,000 days from an inverted-five-spot well pattern with the four producers in the corners operating at constant BHP. We inject a single polymer slug between Days 400 and 800. Because of the interplay between the global flow field with diluted polymer and local variations in petrophysical properties, different regions will experience very different nonlinear behaviors. Buoyancy effects are not included because the single layer is completely horizontal.

Fig. 12 shows permeability and porosity together with water saturation and polymer concentration after 1,000 days. We simulate the same problem with three different timestep selections. The first uses 100 timesteps of 20 days each, the second uses 20 timesteps of 100 days each, and the third consists of only three timesteps: one for the initial waterflood, one for the polymer injection, and one for the period after the polymer slug has been injected, as an extreme test of robustness. We simulate all three setups with the Newton, LS, TR, and ATR solvers, with Newton and LS set to perform 25 iterations before chopping. **Fig. 13** reports the total number of iterations for each phase of the injection schedule, whereas **Fig. 14** reports the cumulative number of iterations.

Again, we observe that TR and ATR do not need to cut timesteps. Furthermore, even when we only consider successful Newton iterations, TR uses fewer iterations for the two longest timesteps, whereas ATR uses fewer iterations for all setups. We also see that LS wastes iterations even with 100 timesteps, performs worse than TR for all setups except the one with 100 timesteps, and worse than ATR for all setups. Comparing successful iterations, LS also requires more iterations than TR for the setups with three and 20 steps, and more iterations than ATR for all setups. Finally, we note that adaptivity reduces the number of iterations to 59% for 100 timesteps and 86% for 20 timesteps compared with static TR, whereas the iteration count increases by 2% with three timesteps. This is in line with our previous discussion of how the ATR method should be configured: For very long timesteps, TRs are needed for most of the iterations and interfaces and should be invoked by default as a precaution. We can also obtain an exact match between TR and ATR if we invoke TRs for more than $n_{\mathrm{osc}} + 1$ iterations when n_{osc} is small (see Algorithm 1).

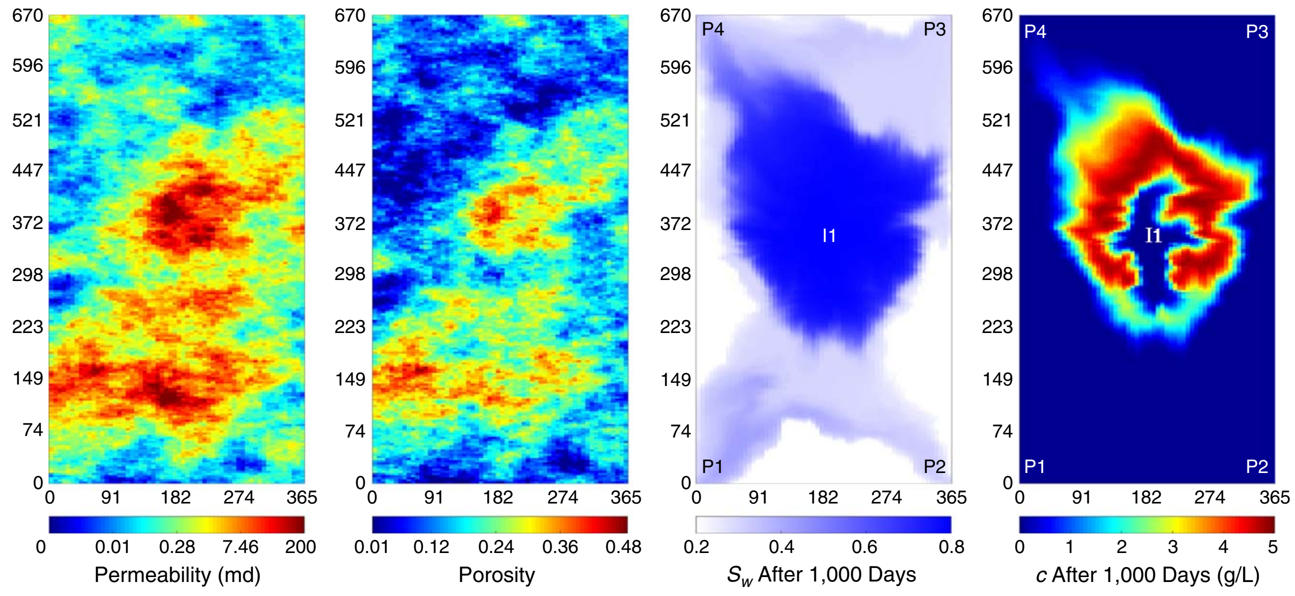

Fig. 12—Petrophysical properties and solution profiles for water saturation and polymer concentration for the SPE 10 subset example.

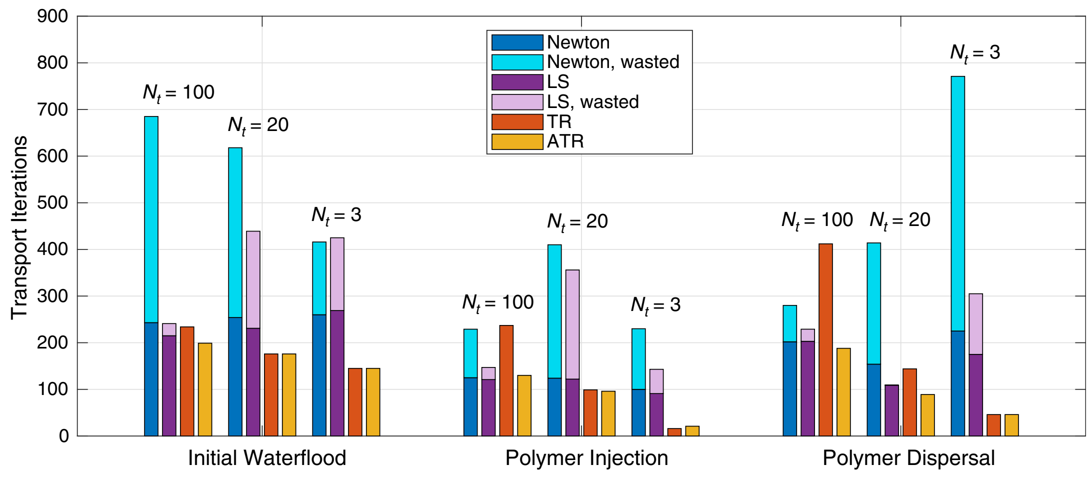

Fig. 13—Total number of iterations used by the transport solvers for the SPE 10 example per solver for each part of the schedule. LS = Newton solver with line search.

Fig. 14 also reports the cumulative number of residual evaluations. With this as a measure of efficiency, ATR clearly outperforms the other solvers, except with three timesteps, for which TR is insignificantly better.

Polymer Example 2: Norne Field Model. Polymer flooding has never been used in the oil/gas Norne Field, and herein we only use parts of the simulation model (Open Porous Media Initiative 2015) to create a difficult test for the four nonlinear solvers. We keep the grid geometry and petrophysical data, except for the three nearly disconnected top layers, giving a model with 37,702 active cells. Instead of using the wells from the real simulation model, we set up a completely artificial well pattern consisting of eight vertical injectors and six vertical producers distributed throughout the whole reservoir volume and completed in all layers. To ensure that the water/polymer mixture sweeps a large portion of the reservoir so that the evolving displacement fronts pass through as much of the heterogeneity and complex cell geometries in the model as possible, we simulate an injection horizon of 75 years with polymer injected between Years 15 and 30 in all injectors. **Fig. 15** shows reservoir geometry and petrophysical properties along with the water saturation and polymer concentration at the end of the simulation period.

Viscosity is 1 cp for pure water and 10 cp for the oil phase. Relative permeabilities are Brooks-Corey with exponent of 3 and residual saturation of 0.15 for oil and exponent of 2 and residual saturation of 0.1 for water. The densities of oil and water 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 200, 100, and 20 uniform timesteps, respectively, and let the Newton and LS solvers perform 25 iterations before chopping. **Figs. 16 and 17** report the total and cumulative number of iterations for the Newton, LS, and ATR solvers during each stage of the injection. Fig. 17 also reports the cumulative number of residual evaluations.

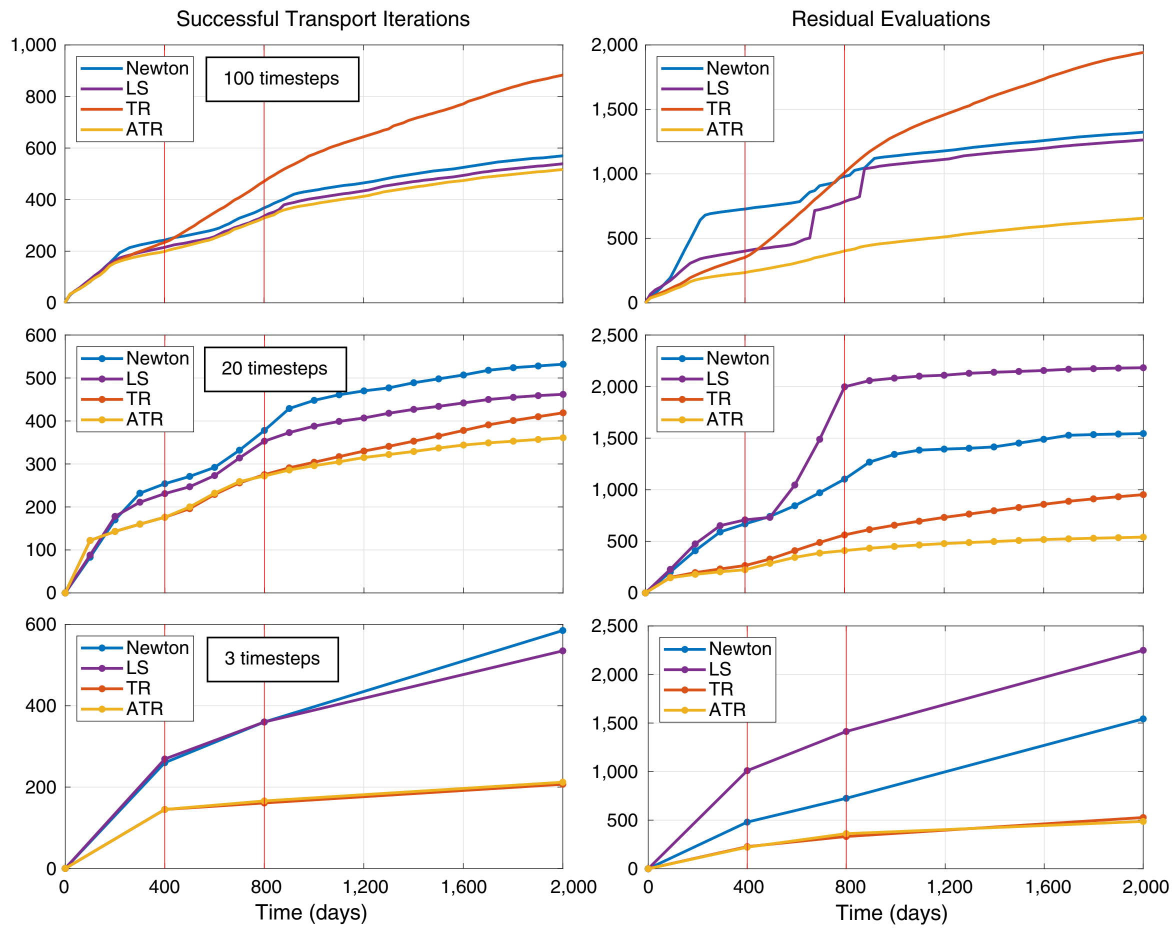

Fig. 14—Cumulative number of successful iterations and residual evaluations as functions of time for the three different schedules in the SPE 10 example. Vertical red lines indicate the polymer-injection period.

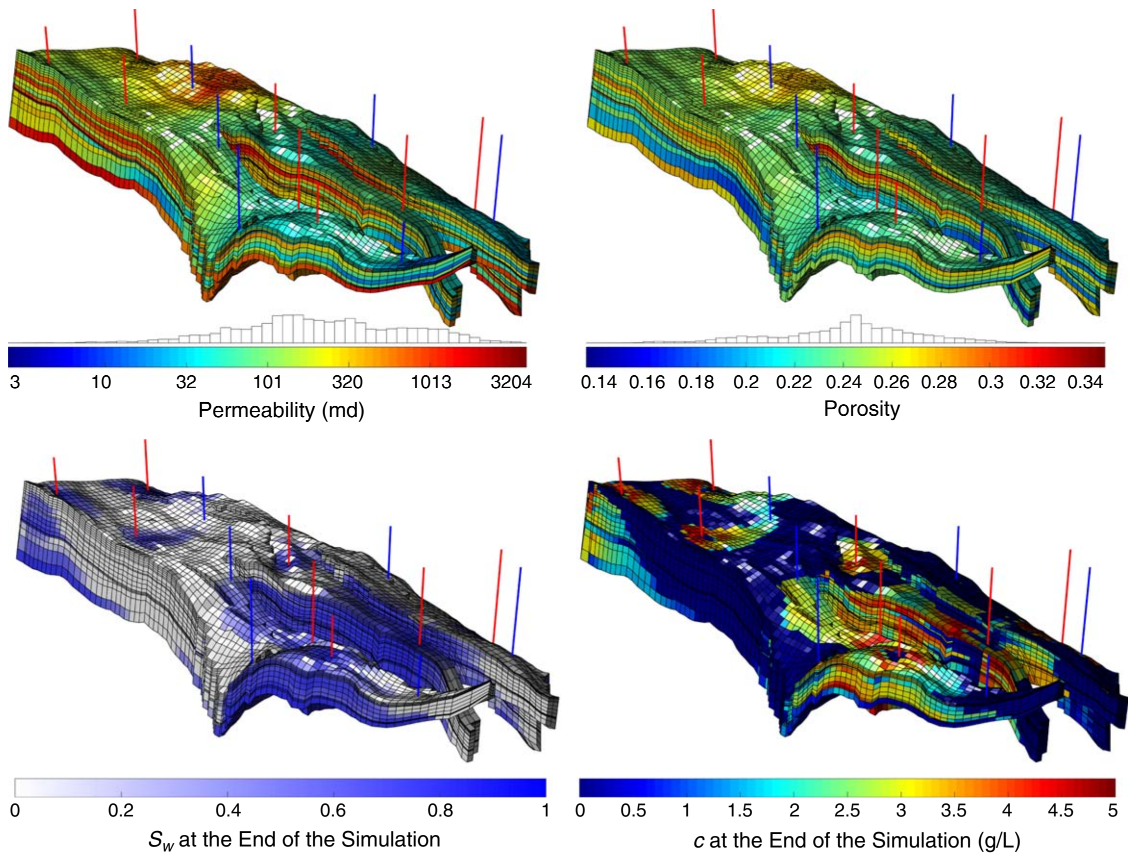

Fig. 15—Petrophysical properties and solution profiles for the Norne Field example.

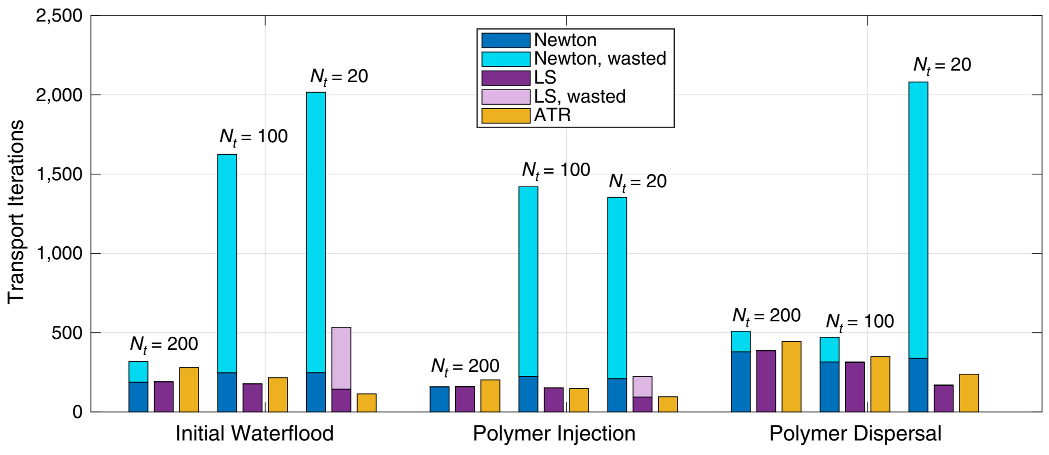

Fig. 16—Total number of iterations per solver for each part of the schedule for the Norne Field example.

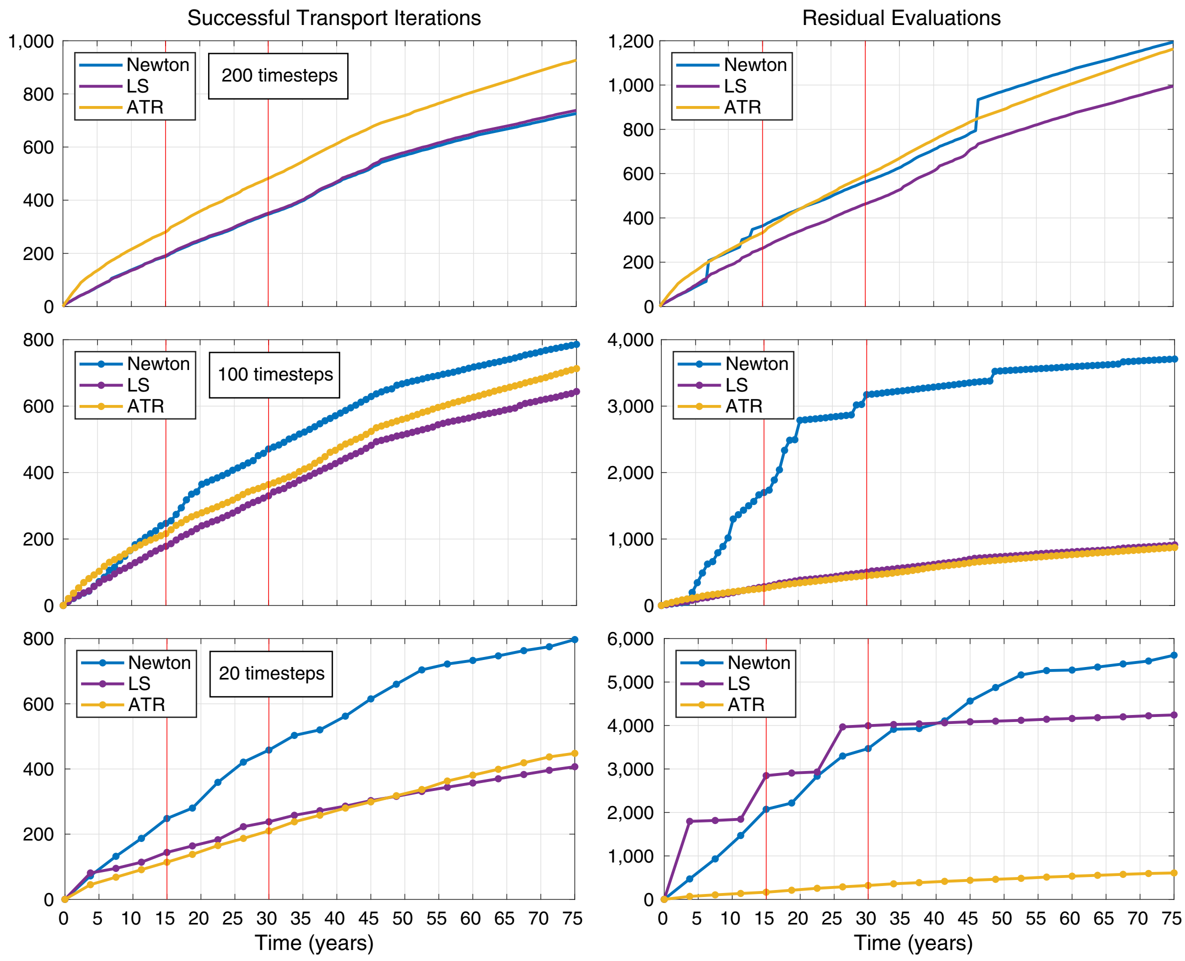

Fig. 17—Cumulative number of successful iterations and residual evaluations as functions of time for the three different setups in the Norne Field example. Vertical red lines indicate the polymer-injection period.

With 200 prescribed timesteps, the standard Newton solver works well during the initial waterflood and the final polymer-dispersion periods and only wastes a moderate number of iterations. During polymer injection, the residual equations become more nonlinear and the number of wasted iterations increases significantly. We can avoid chopping timesteps by using LS and end up with approximately the same number of nonlinear solves as for the successful Newton steps. The ATR solver is more restrictive and requires 13% more iterations than the successful Newton steps. The solver also requires more residual evaluations than LS and slightly fewer than the Newton solver.

With 100 prescribed timesteps, we see a dramatic increase in the number of wasted iterations for the standard Newton solver, but not for the LS solver, which requires fewer iterations than ATR overall and during the first and third parts of the simulation. The number of residual evaluations, however, is slightly fewer for ATR than for LS. With 20 timesteps, Newton has to cut the majority of the steps and hence wastes a large number of iterations. The LS solver also wastes iterations and is overall outperformed by ATR, even though the successful steps of the two Newton solvers consume fewer iterations in total. In terms of residual evaluations, ATR is nevertheless superior to the other two.

The TR solver converges for all timestep lengths, but consumes approximately three times as many iterations as the successful Newton steps (but fewer overall iterations) in the 100-step simulation, and seven times as many (and more iterations overall) for the 200-step simulation. TR also consumes more iterations than LS in both simulations. With 20 timesteps, however, TR outperforms both the Newton and LS solvers. For brevity, neither of these results are reported in Fig. 17.

Convergence issues for the Newton solver are usually a sign that there are dynamics in the reservoir that need to be resolved using smaller timesteps. To show the effect of using TRs to force the solver to converge for the prescribed timestep, **Fig. 18** reports the total oil-production rate from all producers for the 100- and 20-step simulations, with the 200-step solution as a reference. Fig. 18 also reports the number of extra timesteps the Newton solver had to perform to advance the solution the prescribed timestep lengths. The discrepancy between ATR and Newton is remarkably small in the 100-step simulations, although the Newton solver had to chop several timesteps to converge. On the other hand, with almost all the 20 prescribed steps chopped, Newton predicts a production profile that matches the 200-step simulation more accurately than the ATR solver. This should come as no surprise because cutting the timesteps effectively makes the numerical diffusion in the simulation closer to the 100-step case.

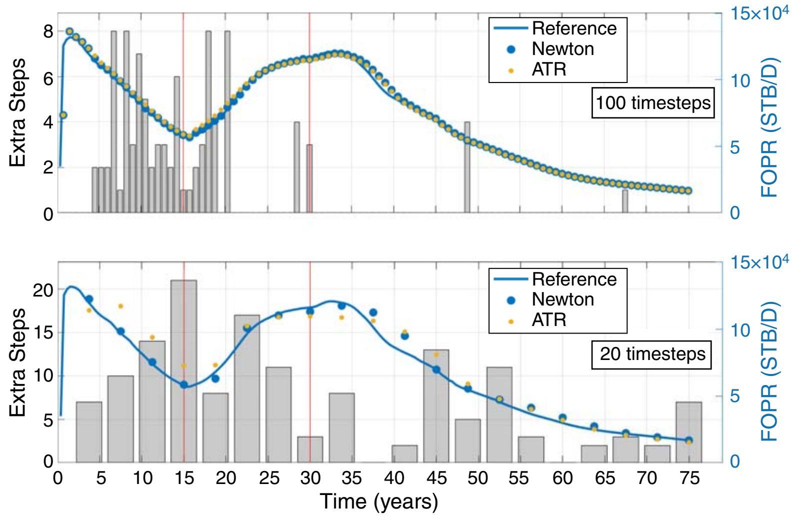

Fig. 18—Production curves for the Norne Field example using Newton and ATR with 100 and 20 target steps, with the 200-step Newton solution as a reference. Although both solvers use the same convergence tolerance, the production curves will deviate because the Newton solver cuts timesteps in the transport solver and thus introduces less numerical diffusion. The bar plot shows the number of extra timesteps needed as a result of the timestep cutting introduced by the Newton solver in each target step. FOPR = field oil-production rate.

Field Model 2: Olympus. The recent ISAPP (2017) optimization challenge is posed on an artificial model inspired by a virgin oil field in the North Sea that spans a lateral area of 9×3 km^2. The reservoir thickness is 50 m and is made up of 16 layers of compressible rock with different petrophysical properties. Permeability and porosity range from 1 md and 0.03 in shale layers to 1,000 md and 0.35 in the channeled sand layers **(Fig. 19).** The model has seven faults: one that makes up the reservoir boundary at one of the sides and six minor internal faults. The computational grid consists of 197,750 cells, each with approximate resolution of $50 \times 50 \times 3$ m^3. The well pattern consists of seven injectors and 11 producers. The fluid model describes a compressible oil/water system with oil and water viscosities of 2.59 and 0.395 cp and densities of 850 and 1020 kg/m^3, respectively. The model has four different facies, each with its own relative permeability curves and residual saturations. The reservoir is initially filled with a mixture of oil and water, and the injectors and producers operate at fixed BHPs of 235 and 150 bar, respectively. Unlike the other test cases in this paper, this constitutes a highly realistic reservoir model.

We use two different setups: one with timesteps starting at 2.8 hours and increasing to the target step length of 30 days during the first eight steps, and one with timesteps staring at 5.6 hours and increasing to 60 days. The Newton and LS solvers are both set to perform a maximum of 25 iterations before chopping the timestep. Fig. 19 shows the initial and final water saturations.

Fig. 20 reports the number of nonlinear transport iterations and residual function evaluations performed by the Newton, LS, and ATR solvers. On the basis of the performance of the TR solver in the previous example, we have chosen not to include it in this example. For a targeted timestep of 60 days, Newton experiences convergence issues in the beginning and wastes a large number of iterations. ATR converges successfully for every timestep and uses approximately 14% more iterations than the successful Newton steps, and approximately one-half as many iterations overall. LS requires approximately the same number of iterations as the successful Newton steps, and overall consumes approximately the same number of residual evaluations as ATR. For a targeted timestep of 30 days, the Newton solver struggles to converge only in one step, and as a result uses approximately 7% fewer iterations than ATR. Again, LS wastes no iterations, and uses approximately the same number of iterations as the successful Newton steps and slightly fewer residual evaluations overall. Water-injection and oil-production rates show insignificant discrepancies between the Newton and ATR methods.

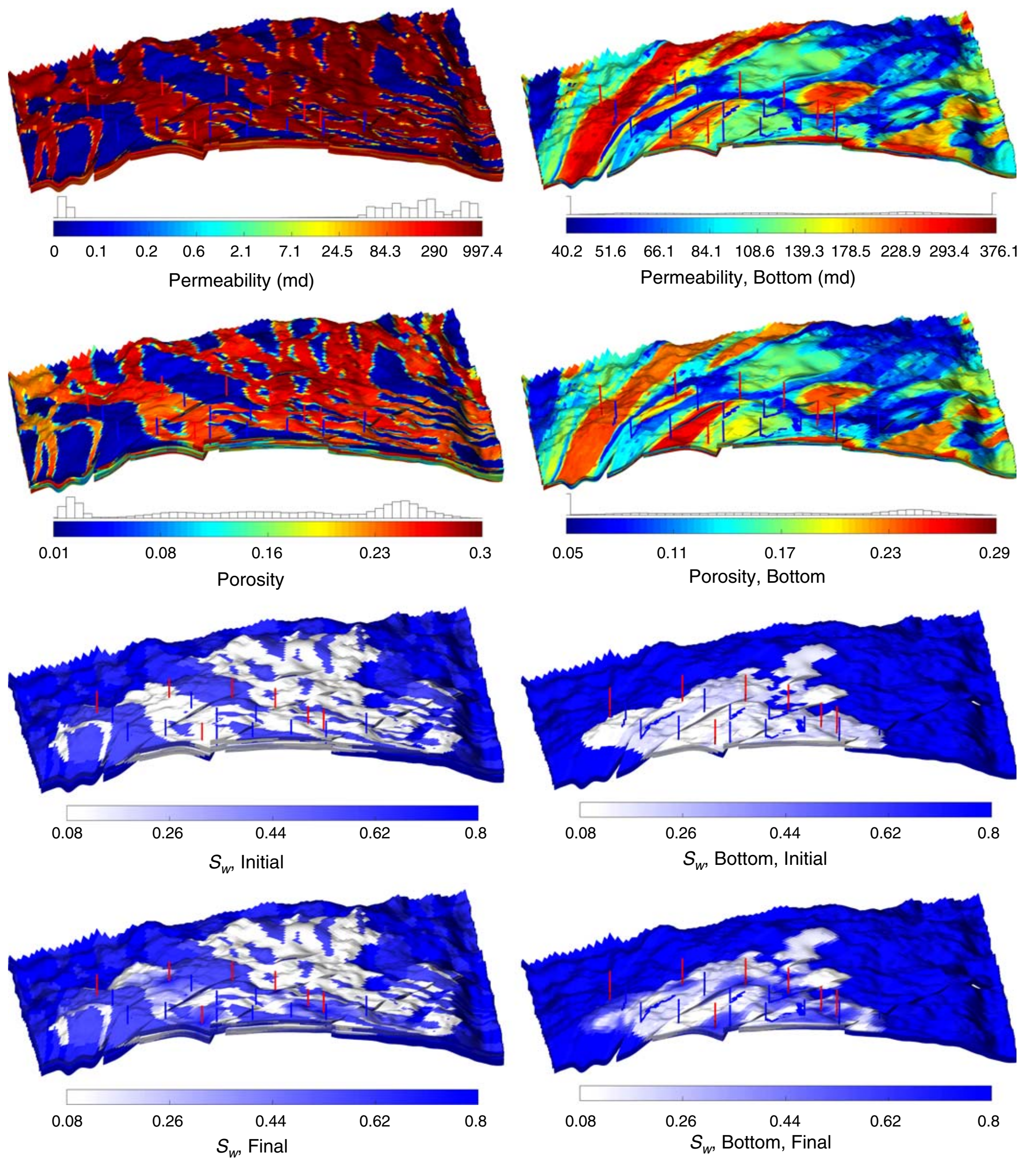

Fig. 19—Permeability, porosity, and initial and final water saturations for the Olympus model. The left column shows the entire model, whereas the right column shows the bottom seven layers. Injectors and producers are shown in red and blue, respectively.

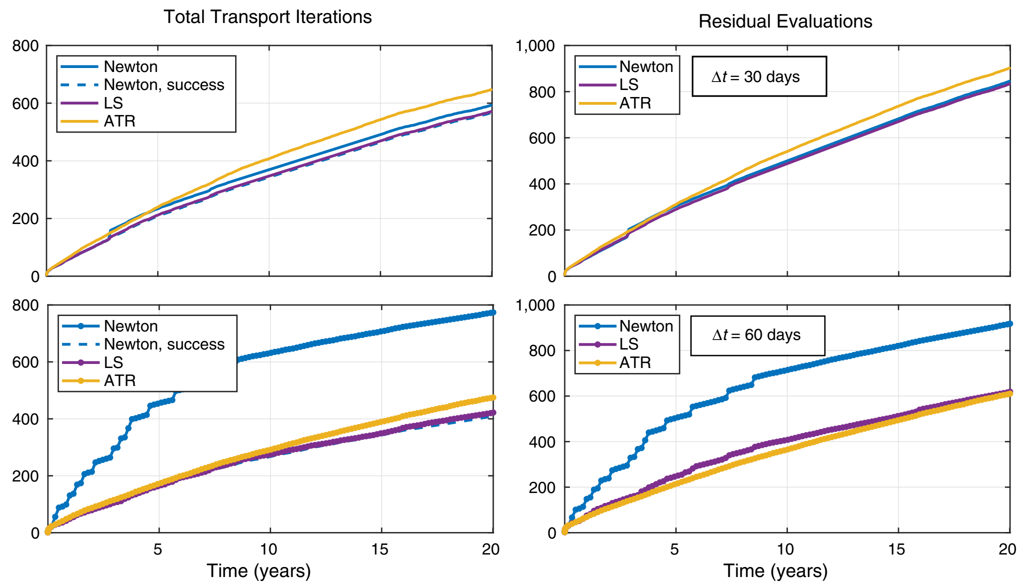

Fig. 20—Cumulative number of iterations and residual evaluations for the Newton, LS, and ATR solvers for the Olympus example.

Conclusions

Summary of Results. We have presented an adaptive algorithm that improves the efficiency of the fully numerical TR method proposed recently by Møyner (2017). The method can be made less restrictive if we only invoke TR checks when we detect oscillations in the Newton updates. A variety of numerical examples demonstrate that our new adaptive method is robust for nearly any timestep, both for contrived test cases designed to stress test the nonlinear solver and for more-realistic simulation scenarios. For reference, we have compared the new solver with the standard optimized Newton solver in MRST. This solver uses the same ad hoc chopping methods as industrial simulators, except for special treatment of transitions to/from mobile/immobile regions, which we disregard to keep the comparison as straightforward as possible. As an option, the Newton solver can use LS to combat convergence issues.

At its best, the Newton solver requires slightly fewer, or equally as many, nonlinear iterations as the ATR solver but is much more sensitive to the choice of timesteps. The number of wasted iterations often increases dramatically if we choose the timestep to be too large in the Newton solvers. This has a strong adverse effect on efficiency and might cause the simulator to slow down and eventually halt completely if the chopping mechanism reduces the timesteps too much. Timestep chopping is particularly evident for cases with strong media contrasts and complex reservoir geometries. These same tendencies are also evident when using Newton with LS, although this solver is not as susceptible to the timestep length.

For the TR solvers, on the other hand, the iteration count per timestep only increased moderately with the timestep in all experiments we have performed (only a few are reported herein). ATR typically uses approximately the same number of iterations as that which the Newton solver consumes in the chopped timesteps. Compared with TR, we see a significant reduction in the number of nonlinear iterations for short to moderately long timesteps, and a close match in iterations for very long timesteps when TRs are active for almost all interfaces throughout the whole simulation. The observed robustness and efficiency of ATR and the efficiency of LS for reasonable timesteps suggest that development of a combined LS/ATR solver is a future possibility.

We also note that the process of finding interface damping factors is inexpensive because it mainly requires us to evaluate relative permeabilities a few extra times, which is significantly less costly than evaluating the full residual. Moreover, most of these evaluations can be performed independently, and we thus believe that the method is well-suited for parallelization.

Future Perspectives. The increased robustness of ATR (or future ATR/LS solvers) can 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, for which standard methods often struggle to converge properly because of large variations in Courant-Friedrichs-Lewy (CFL) numbers between cells and small intercell areas. With the unconditional convergence of TRs, the solver is guaranteed to converge even for cells with very large CFL numbers, thereby providing the robustness necessary to get a simulation through. As such, the method might be attractive as a complement to contemporary multiscale pressure solvers (Lie et al. 2017).

Absolute control of the timestep length is also important to compare different model parameters: Because the TR algorithm guarantees that all models are simulated with the prescribed timesteps, one can disregard differences in temporal discretization errors and attribute observed discrepancies to differences in model parameters. Another example is ensemble simulations, for which it is important that the simulator manages to simulate all ensemble members (with comparable timesteps) to avoid introducing bias in the ensemble averages. A third example is optimization methods, which might perturb the system outside of the parameter region in which the Newton solver works well. The ability to take large timesteps is also beneficial in reduced-physics simulations, in which one might use a single or a few very large timesteps to obtain representative flux fields to quickly estimate how different injection setups affect sweep and displacement efficiency (Krogstad et al. 2017).

Herein, we have only discussed the new ATR method with parameter settings that remain constant throughout the simulation. However, one can easily imagine a more-adaptive algorithm that uses the observed convergence history from previous timesteps to determine whether TRs should be imposed in a cautious or reactive manner. Likewise, the adaptive algorithm has only been applied for global Newton updates that solve for all cells simultaneously. The method can also be combined with more-localized approaches for the transport equations. One such approach is to reduce the fully implicit system by a priori estimation of nonzero update regions before each Newton iteration (Sheth and Younis 2017). Another approach is to reorder the grid cells either according to fluid potential (Kwok and Tchelepi 2007) or by using topological traversal of the interface flux graph (Natvig and Lie 2008) to develop highly efficient iterative Gauss-Seidel solvers that repeatedly sweep through the cells in a predefined order and solve single-cell problems (Lie et al. 2014; Raynaud et al. 2016; Klemetsdal et al. 2018).

Nomenclature

A_i =	discretized accumulation term in cell i
b_α =	shrinkage factor of phase α $[\rho_\alpha = b_\alpha(p_\alpha)\rho_\alpha^s]$, dimensionless
$\boldsymbol{C}$ =	connectivity matrix
$\boldsymbol{d}$ =	normalized Newton update direction
F_{ij} =	discretized flux across interface Γ_{ij}
g =	gravity acceleration, m/s^2
$\boldsymbol{J}$ =	Jacobian matrix of $\boldsymbol{R}$
$k_{r,\alpha}$ =	relative permeability of phase α, dimensionless
$\boldsymbol{K}$ =	permeability, md
n_{chk} =	number of update directions back in time we compare with current in ATR
n_{osc} =	number of observed oscillations over an interface in ATR
n_{TR} =	counter to determine if TR should be invoked over an interface in ATR
N =	number of grid cells
p_α =	pressure of phase α, Pa
P_c =	capillary pressure, Pa
q_α =	sources/sinks of phase α, m^3/s
Q_i =	discretized sources/sinks in cell i
$\boldsymbol{R}$ =	vector of cell residuals
R_q =	residual equation for quantity q
$\boldsymbol{S}$ =	vector of cell saturations
S_α =	saturation of phase α, dimensionless
V_i =	volume of cell i, m^3

$\vec{v}$ =	total macroscopic Darcy velocity $\left(\vec{v} = \sum_\alpha \vec{v}_\alpha\right)$, m/s
$\vec{v}_\alpha$ =	macroscopic Darcy velocity of phase α, m/s
μ_α =	dynamic viscosity of phase α, cp
λ_α =	mobility of phase α ($\lambda_\alpha = k_{r,\alpha}/\mu_\alpha$), cp^{-1}
ρ_α =	density of phase α, kg/m^3
ρ_α^s =	density of phase α at surface conditions, kg/m^3
Γ_{ij} =	common interface of cell i and j
Δt =	timestep, s
$\mathcal{N}(i)$ =	indices of cells sharing common interface with cell i
θ_i =	damping factor for cell i
$\partial_{\boldsymbol{d}}$ =	derivative in direction of Newton update
ϕ =	porosity, dimensionless

Subscripts and Superscripts

i, j =	cell and interface subscripts
n =	timestep
α =	phase subscript
ℓ =	Newton iteration counter

Acknowledgments

Authors Øystein S. Klemetsdal and Knut-Andreas Lie were supported by the Research Council of Norway (Grant No. 244361). Author Olav Møyner is funded by VISTA, which is a basic research program funded by Equinor and conducted in close collaboration with the Norwegian Academy of Science and Letters.

References

Christie, M. A. and Blunt, M. J. 2001. Tenth SPE Comparative Solution Project: A Comparison of Upscaling Techniques. *SPE Res Eval & Eng* **4** (4): 308–317. SPE-72469-PA. https://doi.org/10.2118/72469-PA.

Demidov, D. 2018. AMGCL–A C++ Library for Solving Large Sparse Linear Systems With Algebraic Multigrid Method, March 2018, https://github.com/ddemidov/amgcl (accessed 24 May 2019).

Demidov, D. and Rossi, R. 2017. Subdomain Deflation and Algebraic Multigrid: Combining Multiscale With Multilevel. arXiv preprint arXiv:1710.03940.

Fritsch, F. N. and Carlson, R. E. 1980. Monotone Piecewise Cubic Interpolation. *SIAM J. Numer. Anal.* **17** (2): 238–246. https://doi.org/10.1137/0717021.

Gries, S., Stüben, K., Brown, G. L. et al. 2014. Preconditioning for Efficiently Applying Algebraic Multigrid in Fully Implicit Reservoir Simulations. *SPE J.* **19** (4): 726–736. SPE-163608-PA. https://doi.org/10.2118/163608-PA.

Hilden, S. T., Møyner, O., Lie, K.-A. et al. 2016. Multiscale Simulation of Polymer Flooding With Shear Effects. *Transp Porous Media* **113** (1): 111–135. https://doi.org/10.1007/s11242-016-0682-2.

Integrated Systems Approach for Petroleum Production (ISAPP). 2017. The Olympus Benchmark Case, http://www.isapp2.com/optimization-challenge/reservoir-model-description.html (accessed 24 May 2019).

Jenny, P., Lee, S. H., and Tchelepi, H. A. 2006. Adaptive Fully Implicit Multi-Scale Finite-Volume Method for Multi-Phase Flow and Transport in Heterogeneous Porous Media. *J Comput Phys* **217** (2): 627–641. https://doi.org/10.1016/j.jcp.2006.01.028.

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. https://doi.org/10.1016/j.jcp.2009.06.032.

Jiang, J. and Tchelepi, H. A. 2018. Nonlinear Acceleration of Sequential Fully Implicit (SFI) Method for Coupled Flow and Transport in Porous Media. arXiv preprint arXiv:1810.02326.

Killough, J. E. and Wheeler, M. F. 1987. Parallel Iterative Linear Equation Solvers: An Investigation of Domain Decomposition Algorithms for Reservoir Simulation. Presented at the SPE Symposium on Reservoir Simulation, San Antonio, Texas, 1–4 February. SPE-16021-MS. https://doi.org/10.2118/16021-MS.

Klemetsdal, Ø. S., Rasmussen, A. F., Møyner, O. et al. 2018. Nonlinear Gauss-Seidel Solvers With Higher Order For Black-Oil Models. Presented at ECMOR XVI–16th European Conference on the Mathematics of Oil Recovery, Barcelona, Spain, 3–6 September. https://doi.org/10.3997/2214-4609.201802130.

Krogstad, S., Lie, K.-.A., Møyner, O. et al. 2015. MRST-AD—An Open-Source Framework for Rapid Prototyping and Evaluation of Reservoir Simulation Problems. Presented at the SPE Reservoir Simulation Symposium, Houston, 23–25 February. SPE-173317-MS. https://doi.org/10.2118/173317-MS.

Krogstad, S., Lie, K.-A., Nilsen, H. M. et al. 2017. Efficient Flow Diagnostics Proxies for Polymer Flooding. *Computat Geosci* **21** (5): 1203–1218. https://doi.org/10.1007/s10596-017-9681-9.

Kwok, F. and Tchelepi, H. 2007. Potential-Based Reduced Newton Algorithm for Nonlinear Multiphase Flow in Porous Media. *J Comput Phys* **227** (1): 706–727. https://doi.org/10.1016/j.jcp.2007.08.012.

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. https://doi.org/10.1016/j.egypro.2014.10.395.

Lie, K.-A. 2019. *An Introduction to Reservoir Simulation Using MATLAB/GNU Octave: User Guide for the MATLAB Reservoir Simulation Toolbox (MRST)*. Cambridge, UK: Cambridge University Press.

Lie, K.-A., Nilsen, H. M., Rasmussen, A. F. et al. 2014. Fast Simulation of Polymer Injection in Heavy-Oil Reservoirs on the Basis of Topological Sorting and Sequential Splitting. *SPE J.* **19** (6): 991–1004. SPE-163599-PA. https://doi.org/10.2118/163599-PA.

Lie, K.-A., Møyner, O., Natvig, J. R. et al. 2017. Successful Application of Multiscale Methods in a Real Reservoir Simulator Environment. *Computat Geosci* **21** (5): 981–998. https://doi.org/10.1007/s10596-017-9627-2.

Lu, B., Alshaalan, T., and Wheeler, M. F. 2007. Iteratively Coupled Reservoir Simulation for Multiphase Flow. Presented at the SPE Annual Technical Conference and Exhibition, Anaheim, California, 11–14 November. SPE-110114-MS . https://doi.org/10.2118/110114-MS.

MATLAB is a registered trademark of The MathWorks, Inc., 1 Apple Hill Drive, Natick, Massachusetts 01760-2098.

Moncorgé, A., Tchelepi, H., and Jenny, P. 2017. Modified Sequential Fully Implicit Scheme for Compositional Flow Simulation. *J Comput Phys* **337** (15 May): 98–115. https://doi.org/10.1016/j.jcp.2017.02.032.

Moncorgé, A., Tchelepi, H., and Jenny, P. 2018. Sequential Fully Implicit Formulation for Compositional Simulation Using Natural Variables. *J Comput Phys* **371** (15 October): 690–711. https://doi.org/10.1016/j.jcp.2018.05.048.

Møyner, O. 2017. Nonlinear Solver for Three-Phase Transport Problems Based on Approximate Trust Regions. *Computat Geosci* **21** (5–6): 999–1021. https://doi.org/10.1007/s10596-017-9660-1.

Møyner, O. and Lie, K.-A. 2016. A Multiscale Restriction-Smoothed Basis Method for Compressible Black-Oil Models. *SPE J.* **21** (6): 2079–2096. SPE-173265-PA. https://doi.org/10.2118/173265-PA.

Møyner, O. and Moncorgé, A. 2018. Nonlinear Domain Decomposition Scheme For Sequential Fully Implicit Formulation of Compositional Multiphase Flow. Presented at ECMOR XVI–16th European Conference on the Mathematics of Oil Recovery, Barcelona, Spain, 3–6 September. https://doi.org/10.3997/2214-4609.201802128.

Møyner, O. and Tchelepi, H. 2018. A Mass-Conservative Sequential Implicit Multiscale Method for Isothermal Equation-of-State Compositional Problems. *SPE J.* **23** (6): 2376–2393. SPE-182679-PA. https://doi.org/10.2118/182679-PA.

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. https://doi.org/10.1016/j.jcp.2008.08.024.

Odeh, A. S. 1981. Comparison of Solutions to a Three-Dimensional Black-Oil Reservoir Simulation Problem. *J Pet Technol* **33** (1): 13–25. SPE-9723-PA. https://doi.org/10.2118/9723-PA.

Open Porous Media Initiative. 2015. The Norne Dataset, https://github.com/OPM/opm-data (accessed 24 May 2019).

Raynaud, X., Lie, K.-A., Nilsen, H. M. et al. 2016. The Single-Cell Transport Problem for Two-Phase Flow With Polymer. *Computat Geosci* **20** (3): 495–507. https://doi.org/10.1007/s10596-015-9502-y.

Schlumberger. 2013. *ECLIPSE: Technical Description 2013.2*. Houston: Schlumberger.

Sheth, S. M. and Younis, R. M. 2017. Localized Solvers for General Full-Resolution Implicit Reservoir Simulation. Presented at the SPE Reservoir Simulation Conference, Montgomery, Texas, 20–22 February. SPE-182691-MS. https://doi.org/10.2118/182691-MS.

Tarjan, R. 1972. Depth-First Search and Linear Graph Algorithms. *SIAM J. Comput.* **1** (2): 146–160. https://doi.org/10.1137/0201010.

Todd, M. R. and Longstaff, W. J. 1972. The Development, Testing, and Application of a Numerical Simulator for Predicting Miscible Flood Performance. *J Pet Technol* **24** (7): 874–882. SPE-3484-PA. https://doi.org/10.2118/3484-PA.

Trangenstein, J. A. and Bell, J. B. 1989. Mathematical Structure of the Black-Oil Model for Petroleum Reservoir Simulation. *SIAM J. Appl. Math.* **49** (3): 749–783. https://doi.org/10.1137/0149044.

Trottenberg, U., Oosterlee, C. W., and Schuller, A. 2000. *Multigrid*, first edition. Cambridge, Massachusetts: Academic Press.

Voskov, D. V. and Tchelepi, H. 2011. Compositional Nonlinear Solver Based on Trust Regions of the Flux Function Along Key Tie-Lines. Presented at the SPE Reservoir Simulation Symposium, The Woodlands, Texas, 21–23 February. SPE-141743-MS. https://doi.org/10.2118/141743-MS.

Wallis, J. R. 1983. Incomplete Gaussian Elimination as a Preconditioning for Generalized Conjugate Gradient Acceleration. Presented at the SPE Reservoir Simulation Symposium, San Francisco, 15–18 November. SPE-12265-MS. https://doi.org/10.2118/12265-MS.

Wang, X. and Tchelepi, H. A. 2013. Trust-Region Based Solver for Nonlinear Transport in Heterogeneous Porous Media. *J Comput Phys* **253** (15 November): 114–137. https://doi.org/10.1016/j.jcp.2013.06.041.

Watts, J. 1986. A Compositional Formulation of the Pressure and Saturation Equations. *SPE Res Eng* **1** (3): 243–252. SPE-12244-PA. https://doi.org/10.2118/12244-PA.

Younis, R., Tchelepi, H. A., and Aziz, K. 2010. Adaptively Localized Continuation-Newton Method—Nonlinear Solvers That Converge All the Time. *SPE J.* **15** (2): 526–544. SPE-119147-PA. https://doi.org/10.2118/119147-PA.

Øystein S. Klemetsdal is a PhD-degree candidate at the Norwegian University of Science and Technology (NTNU) and SINTEF Digital. His research interests include discretization methods and linear/nonlinear solvers for flow in porous media, EOR, and scientific software development. Klemetsdal holds a master's degree in applied mathematics from NTNU.

Olav Møyner is a post-doctoral-degree researcher at NTNU and a research scientist at SINTEF Digital. His research interests include multiscale discretizations, linear and nonlinear solvers for flow and transport in porous media, and scientific software development. Møyner holds a PhD degree in applied mathematics from NTNU.

Knut-Andreas Lie is chief scientist at SINTEF Digital and a professor at NTNU. His research interests include gridding and discretization, upscaling and multiscale methods, solution strategies and nonlinear solvers, carbon dioxide storage, and open-source software. Lie holds a PhD degree in applied mathematics from NTNU. He currently serves as executive editor for *SPE Journal*.