Adaptively Localized Continuation-Newton Method—Nonlinear Solvers That Converge All the Time

R.M. Younis, H.A. Tchelepi, and **K. Aziz,** SPE, Stanford University

Summary

Growing interest in understanding, predicting, and controlling advanced oil-recovery methods emphasizes the importance of numerical methods that exploit the nature of the underlying physics. The fully implicit method offers unconditional stability of the discrete approximations. This stability comes at the expense of transferring the inherent physical stiffness onto the coupled nonlinear residual equations that are solved at each timestep. Current reservoir simulators apply safeguarded variants of Newton's method that can neither guarantee convergence nor provide estimates of the relation between convergence rate and timestep size. In practice, timestep chops become necessary and are guided heuristically. With growing complexity, such as in thermally reactive compositional flows, convergence difficulties can lead to substantial losses in computational effort and prohibitively small timesteps. We establish an alternative class of nonlinear iteration that converges and associates a timestep to each iteration. Moreover, the linear solution process within each iteration is performed locally.

By casting the nonlinear residual equations for a given timestep as an initial-value problem, we formulate a continuation-based solution process that associates a timestep size with each iteration. Subsequently, no iterations are wasted and a solution is always attainable. Moreover, we show that the rate of progression is as rapid as that for a convergent standard Newton method. Moreover, by exploiting the local nature of nonlinear wave propagation typical to multiphase-flow problems, we establish a linear solution process that performs computation only where necessary. That is, given a linear convergence tolerance, we identify a minimal subset of solution components that will change by more than the specified tolerance. Using this a priori criterion, each linear step solves a reduced system of equations. Several challenging examples are presented, and the results demonstrate the robustness and computational efficiency of the proposed method.

Introduction

Multiphase, multicomponent flows through subsurface porous media couple several physical phenomena with vastly differing characteristic scales. The fastest processes, such as component-phase equilibria, are assumed to occur instantaneously, and they are modeled as nonlinear algebraic constraints. Mass-conservation laws govern the transport of components propagating through the flow field. These transport phenomena are near-hyperbolic, and they evolve with a finite domain of dependence. Moreover, the flow field itself is transient and evolves with parabolic or elliptic character. The underlying constitutive relations, such as those for the velocity of a fluid phase, couple the variables across the governing equations in a strongly nonlinear manner. Consequently, one challenge in modeling large-scale flows through complex media is to honor such coupling without sacrificing numerical stability. Additional sources of complexity include the heterogeneity of the porous media, body forces, and the presence of wells.

This paper (SPE 119147) was accepted for presentation at the SPE Reservoir Simulation Symposium, The Woodlands, Texas, USA, 2–4 February 2009, and revised for publication. Original manuscript received for review 14 November 2008. Revised manuscript received for review 28 May 2009. Paper peer approved 1 June 2009.

A rich collection of numerical treatments is used in practice to model such problems. For a review, see Aziz and Settari (1979). One approach is to seek methods that are tailored to resolving the physics within distinct regimes. With a priori characterization of how these regimes interact and when they occur, tailored methods can be combined and applied adaptively through the course of a simulation. A second approach is to seek methods that resolve a wide range of processes in a fully coupled manner. Robust nonlinear solvers are particularly important when regime transitions occur frequently and in a complex manner. Strong nonlinear physical coupling across a wide range of time scales poses challenges to both approaches; in the split-and-couple semi-implicit approach, severe restrictions on the timestep size usually arise, and while the fully coupled implicit approach has no stability restrictions, the resulting algebraic nonlinear systems may be difficult to solve (Aziz and Settari 1979).

In fully implicit/fully coupled methods, all primary unknowns are treated implicitly, giving rise to a coupled nonlinear system of discrete equations that must be solved at each timestep. One attractive aspect of this approach is its unconditional stability, which is obtained at the cost of tight nonlinear coupling between parabolic and hyperbolic components. Two practical shortcomings of this approach continue to receive attention from the research community [e.g., Gropp et al. (2001) and Keyes (2002)]. The first is that available solution methods for the discrete nonlinear systems may themselves not be unconditionally convergent. The second aspect is that, regardless of the technical details of the particular solution method, the computational effort required to solve large coupled systems can be significantly larger than that for decoupled, localized computations, such as in a convergent step of a semi-implicit scheme. The practical implication of these two shortcomings is the use of timestep chops. With a try-adapt-try strategy, an attempt to solve for a timestep is made. If that fails within a specified amount of computational effort, the timestep is adapted heuristically and the previous effort is wasted.

This work focuses on exploiting fundamental understanding of implicit methods for oil recovery problems in order to devise nonlinear solution strategies that converge all the time while performing computations only where necessary. More precisely, an iteration is devised so that, for any requested timestep, iterations are performed until either the solution is attained or the maximum allowable number of iterations is reached. In the latter case, the final iterate is a solution to a known positive timestep. The timestepping then continues from this solution. Moreover, the solution algorithm exploits the locality of concentration waves in order to reduce the computational effort required per iteration. Before developing these two contributions of continuation and localization, we review nonlinear solution methods used in current reservoir simulators. Throughout this discussion, we refer to two reservoir simulation problems described in detail in Appendix A. The first problem is a one-dimensional (1D) Buckley-Leverett problem with gravity, and the second is a two-phase compressible flow problem in 2D.

Current State of Nonlinear Solver Technology

Similar to the residual systems of Problems 1 and 2 introduced in Appendix A, general reservoir simulation implicit models result in nonlinear systems that must be solved at each timestep to obtain the new state. Current simulators rely on a fixed-point iteration, such

as a variant of Newton's method, in order to solve these problems [see, for example, Aziz and Settari (1979), Deuflhard (2004), and Ortega and Rheinboldt (1970)]. For general problems, Newton's method is not guaranteed to converge, and it is known to be sensitive to the initial guess, which must be supplied somehow. In most reservoir simulators, the initial guess to the iteration is the old state. For small timestep sizes, this is a good approximation to the new state and, therefore, is likely to be a good starting point for the Newton iteration. For larger timesteps, however, this is less likely to be the case and the iteration may converge too slowly or even diverge. To illustrate why Newton's method may fail in practice, we consider its origin followed by some examples.

Generalized View of Newton's Iteration

At each timestep of an implicit simulation, given the current state, $U^n \in \mathbb{R}^N$, and a target timestep size, $\Delta t > 0$, we seek to obtain the new state, $U^{n+1} \in \mathbb{R}^N$, by solving the following nonlinear residual system:

$$R\left(U^{n+1};\Delta t,U^n\right)=0\,, \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (1)$$

where the residual system of N equations is a unique and consistent mapping, $R:\mathbb{R}^N \to \mathbb{R}^N$. Newton's method generates a sequence of iterates, $\left[U^{n+1}\right]^\nu$, $\nu = 0,1, \ldots$, that hopefully converges to the solution for the target timestep, U^{n+1}. Denoting the Jacobian matrix of the residual with respect to the unknowns as $\mathbf{J}$, this sequence is generated starting from the old state as follows:

$$\begin{aligned}&\left[U^{n+1}\right]^0 = U^n,\\ &\left[U^{n+1}\right]^{\nu+1} = \left[U^{n+1}\right]^\nu - \mathbf{J}^{-1}R\left(\left[U^{n+1}\right]^\nu;\Delta t,U^n\right), \quad \nu = 0,1,\ldots .\end{aligned}$$
$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (2)$$

Newton's iteration itself can be regarded as an explicit, first-order timestepping scheme for a particular dynamic system. To see this, suppose that the Newton iteration index, ν, is actually a continuous quantity and consider the following dynamic system:

$$\begin{aligned}&U^{n+1} = U^n, && \nu = 0\\ &\frac{dU^{n+1}}{d\nu} = -J^{-1}R\left(U^{n+1};\Delta t,U^n\right), && \nu > 0\,. \end{aligned} \quad \ldots\ldots\ldots\ldots\ldots\ldots (3)$$

Using a first-order, explicit discretization in the continuous quantity ν, the discrete form of the dynamic system is

$$\left[U^{n+1}\right]^{\nu+1} - \left[U^{n+1}\right]^\nu = -\Delta\nu\mathbf{J}^{-1}R\left(\left[U^{n+1}\right]^\nu;\Delta t,U^n\right), \quad \ldots\ldots\ldots (4)$$

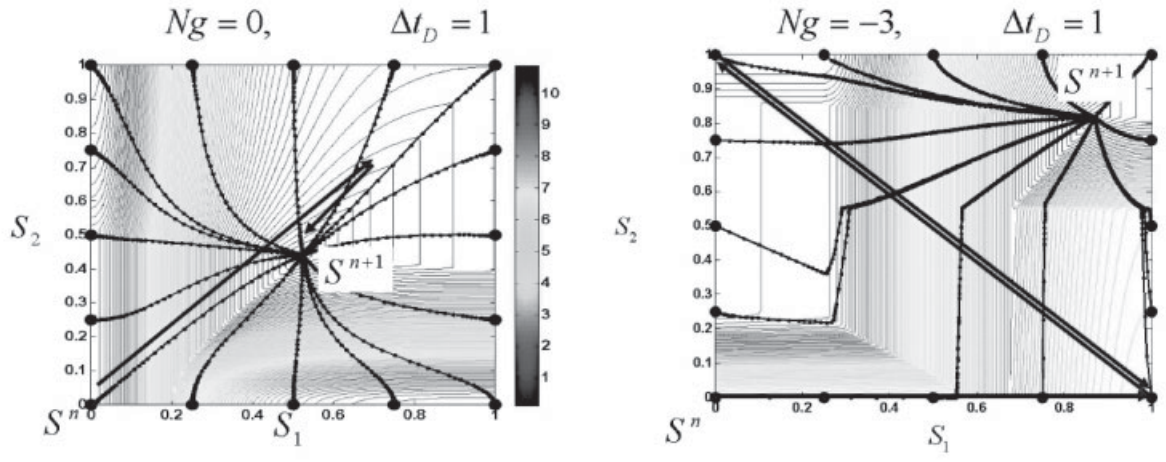
which corresponds to Newton's method (Eq. 2). Thus, Newton's method approximates the derivative of the new state with respect to the embedded time, ν, using a first-order finite difference with a unit step size, $\Delta\nu = 1$. The forcing function of the dynamical system (Eq. 3) is the inverse of the Jacobian acting on the residual evaluated at the old embedded step, ν. Note that the embedded time, ν, is different from the physical timestep, Δt, which is fixed throughout Newton's iteration. Because the forcing function is evaluated at the old embedded step, Newton's iteration is an explicit first-order discretization of the system in Eq. 3. The embedded timestep, $\Delta\nu$, is a step along the field defined by the system of Eq. 3, which we refer to as the Newton flow. At the solution of the target timestep, the forcing function is zero, so the solution is a stable fixed point of the Newton flow. Because explicit first-order timestepping may be unstable (has a timestep restriction), Newton's iteration may not converge, even though the continuous Newton flow is well-behaved. On the other hand, for general problems, when Newton's method does converge, there are no guarantees on how fast it will do so.

In practice, a convergent Newton iteration for a given timestep size may be too slow, and it becomes necessary to stop if a prescribed maximum number of iterations is reached before convergence is achieved. In such cases, the iterates are discarded and the process is repeated using a smaller target timestep. It is not known rigorously which smaller timestep could be used with success. Several heuristics are often employed to attempt this; nevertheless, all such methods fall into a strategy of try, adapt, and try again. It is difficult to derive rigorously a relation between timestep size and convergence rate because that would ultimately involve a stability analysis of the system in Eq. 3, which is both highly problem-dependent and analytically intractable for realistic problems. Moreover, such an analysis is likely to generate iterations that needlessly follow the Newton flow too closely, resulting in a highly inefficient iteration.

Examples Illustrating Challenges Encountered by Newton's Method

To illustrate the pathologies that Newton's method may run into in practice, we consider examples of Problem 1, which is described in Appendix A. We apply a fully implicit discretization on a discrete domain with two cells, $N = 2$. The corresponding residual is regarded as a function in 2D saturation space. The first dimension is the range of possible saturations in the first cell $S_1 \in [0,1]$, and the second dimension is the range in the second cell $S_2 \in [0,1]$. We compare the Newton iterations with the Newton flow in this 2D space for two cases. The first is a horizontal case ($Ng = 0$), and the second is a down-dip problem ($Ng = -3$). In both cases, we use a unit injection saturation, $S_{inj} = 1$, and a uniform zero initial saturation, $S_{init} = 0$.

Figs. 1a and 1b illustrate a Newton process (series of straight, thick blue arrows) for both cases starting from the old state $S^n = (0,$



(a) Horizontal displacement $Ng = 0$ (b) Down-dip displacement $Ng = -3$

Fig. 1—Residual norm contour lines, high-fidelity Newton flow integral paths (dotted), and Newton iterations (arrows) for two cases of Problem 1.

0) for a timestep of unity. The contour lines on the figures are those of the 2-norm of the residual, $\left\|R\left(S;\Delta t_D,S^n\right)\right\|_2$ over $S\in[0,1]\times[0,1]$. The thick, curved black lines are integral numerical solutions of Eq. 3 (i.e., the Newton flow) emanating from various starting points on the boundary of the 2D space. These integral curves are obtained using a high-fidelity numerical integrator (variable-order Runge-Kutta with automatic step control), and they illustrate the continuous paths that Newton's method attempts to approximate should it have been started from these various initial guesses.

For the horizontal case, Fig. 1a shows that Newton's method converges to the solution S^{n+1} = (0.5, 0.45) in two iterations. Moreover, the Newton-flow integral curves are generally smooth curves, all of which flow toward the solution. Fig. 1b, which is for the down-dip case, indicates that Newton's method diverges when started from the (0, 0) point, with the iterates alternating between the points (1, 0) and (0, 1). For this case, the Newton-flow integral curves are not smooth, and, for saturation loci where the upwind directions change, there are clear kinks along the Newton integral curves. The smooth region around the solution, (0.83, 0.9), is considerably smaller than that for the horizontal case in Fig. 1a. In theory, while following the Newton flow more accurately can lead to increased robustness, the computational efficiency of doing so may be unjustifiable. In order to manage this tradeoff between following the Newton flow and arriving as quickly as possible at the solution for the target timestep, several variants of Newton's method have been devised. These variants attempt to safeguard the iteration by improving its ability to follow the Newton flow in some sense.

Safeguarded Newton Iteration

Given the possibility of divergence of Newton's method for general problems, a number of variants have been devised to damp the Newton updates. All of these methods, which are said to safeguard the iteration, can be viewed as different ways to specify a diagonal matrix $\mathbf{\Lambda}=\mathrm{diag}\left(\Delta\nu_1,\ldots,\Delta\nu_N\right)$ in a safeguarded Newton iteration, which we write in the general form as

$$\left[U^{n+1}\right]^{\nu+1}=\left[U^{n+1}\right]^{\nu}-\mathbf{\Lambda}\mathbf{J}^{-1}R\left(\left[U^{n+1}\right]^{\nu};\Delta t,U^n\right). \quad \ldots\ldots\ldots\ldots (5)$$

Comparing the general safeguarded iteration (Eq. 5) with the classic Newton iteration (Eq. 2), the diagonal weights, $\Delta\nu_i$, can be interpreted as local timesteps in the explicit integration of the Newton flow. The standard Newton iteration selects all of these weights to be unity, and subsequently, in cases where the underlying Newton flow changes rapidly, the iteration may not converge. Two classic approaches to safeguard Newton's method are the line-search and trust-region algorithms. See Ortega and Rheinboldt (1970) and Deuflhard (2004) for examples. In these methods, all entries of the diagonal are identical, implying that the Newton direction is simply scaled by a constant factor. The choice of the scaling factor is dictated by the rate of change in the residual norm along the Newton direction or within a neighborhood about the current iterate.

In commercial reservoir simulators, heuristic strategies have also been devised to safeguard Newton's method. Such strategies select the diagonal scaling entries on a cell-by-cell basis using physical arguments. In practice, these heuristics are known to improve convergence (larger timestep sizes can be converged from an initial state). Most commonly employed heuristics share a hypothesis, which may be motivated by the residual contours in Fig. 1 and the shape of common fractional flow curves (e.g., Fig. A-2). The hypothesis is that the nonlinearity of the residual in some sense is dictated by the structure of the flux function in each cell (Jenny et al. 2009). That is, suppose that a nonlinear Gauss-Seidel iteration is applied to the residual system as a whole. Then, for each Gauss-Seidel iteration, the residual in each cell must be solved sequentially. To solve each of these single cell nonlinear residuals, we may apply a scalar Newton process, obtaining the saturation in the current cell, assuming all other cell saturations are known. These scalar Newton iterations need to be safeguarded by some scalar damping protocol (Kwok and Tchelepi 2007). If this protocol ensures cellwise convergence of a Newton iteration, then the outer Gauss-Seidel iteration at least has a hope of converging. So, if the same scalar damping protocols are also applied to the saturation variables on a cell-by-cell basis in the context of system Newton updates, then that, too, is more likely to converge. Inspecting Fig. A-2 and Fig. 1 from the preceding section, we can deduce that saturations around endpoints and inflection points produce particularly sensitive Newton directions. The idea is to apply a cell-by-cell (diagonal) damping factor to limit large changes around such sensitive physics boundaries.

The following is a sample list of heuristic scalar damping protocols for Newton methods in black-oil simulation (Naccache 1997; Schlumberger 2008):

- Eclipse Appleyard (EA): For saturations only, cell by cell, scale back changes from immobile to mobile so that they are barely mobile. For changes from mobile to immobile, scale back to barely mobile. Ensure saturations are between 0 and 1.
- Modified Appleyard (MA): Do the same as in Method 1, and require that no saturation change in a cell is greater than some small amount in magnitude, which is usually chosen to be 0.2.
- Geometric penalty (GP): Do the same as Method 1, and require that no saturation change in a cell is greater than 20% of the original saturation.

The focus of these cellwise strategies is to avoid large Newton corrections to saturation variables, as may arise when crossing phase boundaries, or other pathological properties of the fractional flow curve. Next, we examine practical examples of the behavior of these methods.

State-of-the-Art Solvers

We observe empirical evidence over a suite of numerical experiments conducted using cases of Problem 2, which is described in Appendix A. For various cases of imposed well conditions and initial saturation distributions, the following experiment is performed. A sequence of timestep sizes is fixed, and, for each timestep size, the residual is solved using each of four protocols: Standard Newton (SN), EA, MA, and GP. Each target timestep is solved using the old state as a starting guess. The number of iterations is recorded, and the convergence characteristics are reported. Note that these results correspond to using each of the four methods to solve a full simulation in one timestep.

In a representative case, the initial condition has oil on the lower half of the domain and water above. A rate-controlled water injector is applied in the lower left corner, and a pressure-control producer is placed in the upper right corner.

Fig. 2 shows the iteration count profiles obtained. As is typical of problems involving two or more physical driving forces, SN and EA diverge (do not converge no matter how many iterations are spent) for timesteps larger than 0.5 days. The MA and GP strategies are convergent over the entire tested range of step sizes. The key to this discussion is that, while strategies such as MA and GP lead to convergent iterations for this setting, the number of iterations required grows with the requested timestep size. In practice, a maximum number of iterations is imposed by the user for efficiency considerations. So, while MA may well converge in 150 iterations, if the maximum allowable is 10, then the iteration is halted. The 10 iterations are wasted, and a heuristic must be used to scale back the timestep size to one for which MA may converge in less than 10 iterations. It is this strategy of try, adapt, and try again that often brings a simulation of a complex model with several physical transitions to a halt.

Objectives

Our approach develops two ideas to address nonlinear solver issues in general-purpose reservoir simulation. The first idea devises an iterative solution process for which each iterate is a solution to a timestep that is smaller than the target step. The implications of this are that (1) convergence is always guaranteed, (2) timestep chops do not involve wasted computation, and (3) timestep selection for convergence considerations is no longer necessary. The second idea addresses the fact that solutions to larger timesteps

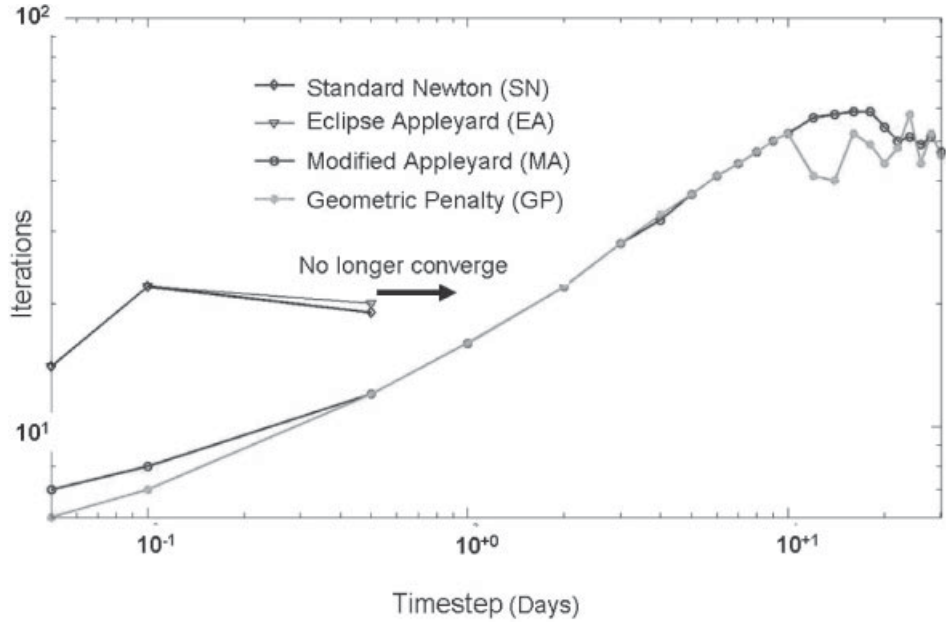


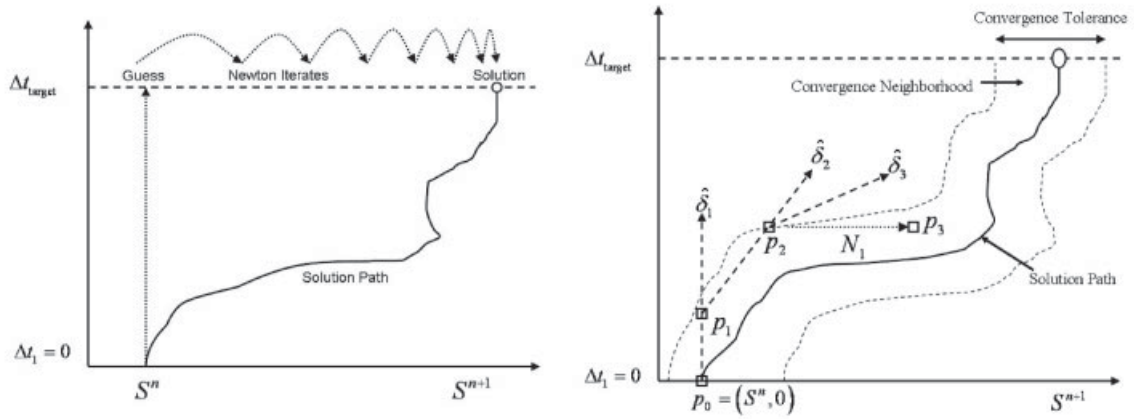
Fig. 2—The number of iterations required to solve a sample timestep size using SN, EA, MA, and GP.

may require more iterations. Exploiting the fact that most reservoir simulation problems involve traveling waves, the second idea, allows each iteration to involve considerably lower computational cost than would be necessary for a standard Newton iteration. It is a method of localizing computation.

Associating a Timestep Size to Iterates—A Continuation Algorithm on Timestep Size

To illustrate our approach, we consider the nature of solutions to reservoir simulation timesteps. Well-posed reservoir simulation residuals have a unique solution for each timestep. The solution needs to be unique only within a prescribed domain. For example, in a cell, saturations are bounded by zero and unity; pressure is positive, etc. Let $U_0 \in \mathbb{R}^N$ denote such a discrete solution state vector for a particular time. Then, independently for every possible timestep from this state, $\Delta t \geq 0$, there corresponds a solution $U \in \mathbb{R}^N$ that satisfies $R(U, \Delta t; U_0) = 0$, where R is the vector of discrete residual equations. All such pairs of solution and corresponding timestep, $(U, \Delta t)$, can be interpreted as points defining a curve in an N+1-dimensional space, where N is the number of residual equations and the additional dimension is the timestep size, Δt. The existence and uniqueness of these points can be shown to imply that they form a continuous curve, emanating from the initial point with a timestep size of zero, $(U_0, 0)$. Furthermore, this curve has a strictly positive gradient along the timestep dimension because otherwise uniqueness would be violated. The curve of points satisfying these requirements for a particular initial state, U_0, is called the solution path emanating from that state.

For illustration, consider the solution path of a nonlinear problem in a single state unknown, that is $R(S^{n+1}, \Delta t; S^n)$. Because only one state is unknown, we may plot the solution path in the two dimensions, S^{n+1} and Δt. In a typical reservoir simulation problem, there may be millions of unknowns, and each would be represented by a dimension. **Fig. 3a** illustrates a solution path to this hypothetical problem. The solution path emanates from the initial point ($S^{n+1} = S^n$, $\Delta t = 0$), and continues to the target timestep,



(a) Newton's method illustrated on a solution path. All iterates are evaluated at the target timestep, starting from the old state as an initial guess.

(b) Illustration of three iterates in the CN algorithm. The first two iterates are chosen along tangent vectors. The third tangent leads to points outside the convergence tolerance. A safeguard Newton step is performed.

Fig. 3—Solution-path diagrams and solution methods depicted for a problem with a single time-dependent unknown S. The old state, for a zero timestep, $\Delta t = 0$, is S^n, and the solution at the target timestep, Δt_{target}, is S^{n+1}.

Δt_{target}, augmented with its solution, S^{n+1}, in this 2D space. Notice that, in this illustration, the solution path always moves forward in timestep size and never folds on itself. Each point on the path is a pair of a solution state and a timestep size.

Also shown in Fig. 3a are the standard Newton steps (curved dotted lines), which attempt to find the solution at the target timestep. With a classic Newton process, the old state is projected to the target time $t_n+\Delta t_{target}$ as an initial guess. A sequence of iterates is obtained, all evaluated using the target timestep size. A convergent Newton iteration provides a final iterate that is close enough to the solution path at the target timestep size. Should too many iterations go by without convergence, all iterates are wasted and the challenging task of selecting a smaller timestep must be addressed.

In order to overcome this challenge, we design a numerical continuation algorithm (Allgower and Georg 1980, 2003; Shroff and Keller 1993; Watson et al. 1987) that proceeds from the initial state from which the solution path emanates. The process generates a sequence of iterates along the path in the augmented N+1-dimensional space, climbing toward the target timestep size. Subsequently, each iterate would be related to a known, smaller timestep. Should too many iterations go by before the target timestep is attained, the final iterate is associated with a solution for a smaller known timestep, from which we may continue the simulation.

High-Level View of the Continuation-Newton (CN) Algorithm

We develop an alternative iteration to Newton's method in which the iterates are pairs of the unknown state augmented with a corresponding timestep. The iteration starts from an initial point consisting of the initial state and a timestep of zero. The intent of the iteration sequence is to follow along the solution path toward the solution of the target timestep size. Because we are interested in the solution for a target timestep, we hope to avoid following the solution path (in the N+1-dimensional space) too closely because that would be computationally wasteful. For this reason, we develop a convergence neighborhood around the solution path so that points within the neighborhood are deemed close enough to the solution path at their timestep level. The iterates can follow the path more loosely without sacrificing their proximity to the solution for their timestep component. By parameterizing the solution path with a single parameter along its tangential coordinates, we develop an ability to compute the tangent to the solution path at any point in the augmented space. These tangent computations are developed in detail in the next section.

Here, we outline the CN process using the single-unknown hypothetical example discussed in the preceding section: $R(S^{n+1}, \Delta t; S^n)$.

Fig. 3b illustrates the proposed algorithm pictorially on the hypothetical solution path similar to that in Fig. 3a. The solution path emanates from the initial point $p_0 = (S^n, 0)$. We want to find the solution for a target timestep Δt_{target}. The proposed algorithm defines a convergence neighborhood around the solution path so that points, $p_{int} = (S_{int}, \Delta t_{int})$, inside the neighborhood are deemed to be close enough solutions for their timestep component, Δt_{int}. That is, either S_{int} is a solution to a timestep of Δt_{int} or it is a good starting guess for an SN iteration starting from S_{int} for a timestep of Δt_{int}. Starting from the initial point, p_0, from which the solution path emanates, we can compute the N+1-dimensional tangent vector, $\hat{\delta}_1$. An appropriate step length along the tangent vector is selected such that the next iterate remains within the convergence neighborhood. In Fig. 3b, this iterate is denoted p_1. The tangent update is linear, $p_1 = p_0 + \alpha\hat{\delta}_1$, where $\alpha \geq 0$ is called the tangent step length. Note that because tangents $\hat{\delta}_i$ and all iterates p_i are augmented variables in both the state and the timestep size, these iterations evolve timestep size as well as the solution. Similarly, we may repeat this process to obtain the next point p_2. At this point, which is near the boundary of the convergence neighborhood, we find that the next tangent step length, which keeps the next iterate within the convergence neighborhood, is too small or zero. In this case, the CN algorithm applies a Newton correction step, N_1, as illustrated in Fig. 3b. This Newton correction is evaluated at the current timestep size and brings the iterates closer to the solution path at point p_3. The iterate, p_3, is guaranteed to be close to the solution path because the convergence neighborhood is chosen so that it is contained within the contraction region of Newton's method. We can continue with tangent steps thereon.

By construction, the sequence of iterates, p_i, are all close-enough solutions to their associated timesteps. The guarantee of convergence for all times is also supported by the choice of the convergence neighborhood. As a result, any necessary corrective Newton steps, N_j, always contract toward the solution path. Because the initial guess to such corrective steps is already a close-enough solution by construction, a single Newton step can be guaranteed to reduce the residual.

In practice, the CN process can be continued until either the target timestep size is attained or the maximal number of iterations allowed has been expended. In the latter case, the final CN iterate is a close-enough solution to its corresponding timestep size. We can accept this solution and its timestep size and continue onto the next simulation timestep. Subsequently, no timestep-chop selection is ever necessary and no computational effort is wasted.

We provide the details of the proposed CN algorithm in Appendix B. Next, we develop the methods to compute tangent updates and to select step lengths that lie within a convergence neighborhood.

Computing Tangents to Solution Paths

The key computational expense of a CN iteration is in evaluating the tangent vectors to the solution path. The cost is precisely that of performing a single Newton iteration and involves the solution of the Jacobian matrix. We derive this computational method by first mathematically parameterizing solution paths along one tangential coordinate, $\lambda \geq 0$. By substituting this parameterization into the residual and requiring a zero residual along the curve, we derive the equations of motion that define the curve. These equations then provide a computationally attractive method for computing the tangent.

The Equations of Motion Along a Solution Path. With the assumption that the N-dimensional residual equations have a unique solution for every timestep, we can parameterize the solution and timestep pairs along a single scalar $\lambda \geq 0$. That is, the N-dimensional state solution vector and corresponding timestep form an N+1-dimensional space, and the components are considered as functions of a single scalar, λ, so that any point can be written as $p(\lambda) = [U(\lambda), \Delta t(\lambda)]$. We want the solution path in the (U, Δt) space to emanate from the old state of a given timestep. So for $\lambda = 0$, we have the point $p_0 = (U_0, 0)$, where U_0 is the initial state vector of the timestep. We write the residual equations in terms of this parameterization as

$$R\left[U(\lambda), \Delta t(\lambda); U^n\right] = 0. \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (6)$$

The total derivative of the residual vector with respect to the scalar λ prescribes the tangent to level curves in the residual value. Because the particular level curve of interest is the zero residual level curve (the solution path), we can write

$$0 = \frac{dR}{d\lambda} = J\frac{dU}{d\lambda} + \frac{\partial R}{\partial \Delta t}\frac{d\Delta t}{d\lambda}, \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (7)$$

where J is the same $N \times N$ Jacobian matrix that would have been used in a Newton method except that it is now interpreted as a function of timestep as well as of state. That is, while within a Newton iteration, the Jacobian is always evaluated using the target timestep, here the Jacobian may be evaluated using any other timestep size. The Jacobian's elements are defined as $J_{(i,j)} = \frac{\partial R_{(i)}}{\partial U_{(j)}}$, and the N-dimensional vector $\frac{\partial R}{\partial \Delta t}$ in Eq. 7 is the derivative of the residual vector with respect to timestep.

Because the initial point on the zero level curve is known (i.e., the solution from the previous timestep), the initial condition for

the system is well defined. Combining this initial condition with Eq. 7, we obtain

$$\begin{cases} J\left(U,\Delta t;U_0\right)\dfrac{dU}{d\lambda}+\dfrac{\partial}{\partial\Delta t}R\left(U,\Delta t;U_0\right)\dfrac{d\Delta t}{d\lambda}=0 \\ U(\lambda=0)=U_0 \\ \Delta t(\lambda=0)=0 \end{cases} \quad \ldots\ldots\ldots\ldots (8)$$

Eq. 8 describes the dynamics of motion along the solution path in (U, Δt) space emanating from the initial point (U_0, 0). In this context, the parameter $\lambda \geq 0$ is the arc length along the solution path in its tangential coordinates. Geometrically, the N+1-dimensional vector $\delta=\left(\dfrac{dU}{d\lambda},\dfrac{d\Delta t}{d\lambda}\right)$ is the tangent to the residual level curves. When the tangent is evaluated at a point on the solution path, which is the zero residual level curve, it is a tangent to the solution path itself.

The solution of a nonlinear residual for any timestep can be computed by numerically integrating the corresponding initial-value problem given by Eq. 8 up to $\lambda=\lambda^*$ for which the timestep is equal to the target value [i.e., $\Delta t(\lambda^*)=\Delta t_{\text{target}}$]. This, however, is not the objective. We are interested in reaching the solution for the target timestep as quickly as possible. Consequently, we should follow the solution path in (U, Δt) space only loosely and with as little computational effort as possible.

Closure and Tangent Computations. Eq. 8 is underdetermined; we have N equations for N+1 unknowns. The additional unknown is the timestep size. Thus, an additional equation is necessary to close the system. Without further insight into the problem, we could choose one of several alternatives to close the system as long as the resulting problem is consistent. One example is to require that the norm of the tangent be unity [see, for example, Allgower and Georg (1980, 2003) and Shroff and Keller (1993)]; that is, $\left\|\left(\dfrac{dU}{d\lambda},\dfrac{d\Delta t}{d\lambda}\right)\right\|=1$. Another is to enforce some sort of inequality constraint on some elements of the tangent. Not only are such alternatives clearly computationally undesirable, but they also do not exploit any inherent properties of the physics. In this particular setting, the additional equation could be chosen to say something about the rate of change in timestep size with respect to arc length along the solution path. Something certain about solution paths to well-posed problems is that $\dfrac{d\Delta t}{d\lambda}>0$. This is the case because otherwise uniqueness is violated.

Enforcing this condition algebraically is quite simple. We require that $\dfrac{d\Delta t}{d\lambda}=C>0$, where C is any positive constant. To compute the N+1-dimensional tangent vector, we solve

$$\begin{bmatrix} J & \dfrac{\partial R}{\partial\Delta t} \\ 0 & 1 \end{bmatrix}\begin{bmatrix} \dfrac{dU}{d\lambda} \\ \dfrac{d\Delta t}{d\lambda} \end{bmatrix}=\begin{bmatrix} 0 \\ C \end{bmatrix}, \quad \ldots\ldots\ldots\ldots\ldots\ldots (9)$$

where the Jacobian, J, is an $N\times N$ matrix; the timestep derivative of the residual, $\dfrac{\partial R}{\partial\Delta t}$, is an N-dimensional vector; and the tangent component along timestep size, $\dfrac{d\Delta t}{d\lambda}$, is a scalar. Notice that the (N+1)×(N+1) system in Eq. 9 is always solvable because the Jacobian itself is assumed to be always solvable. The Jacobian is well conditioned in the vicinity of the solution path. This is because all solution points are stable fixed points of the Newton flow for a fixed timestep and, moreover, we already know that the solution to the timestep size tangent component, $\dfrac{d\Delta t}{d\lambda}$, is $C>0$.

Scaling the Tangent Does Not Require Magic Constants. The positive constant, C, is chosen arbitrarily. The specific choice is inconsequential because we have the unique, consistent direction of the tangent vector and we can select any length along it. To be precise, we can normalize the computed tangent, $\delta=\left(\dfrac{dU}{d\lambda},\dfrac{d\Delta t}{d\lambda}\right)$, as follows:

$$\delta\leftarrow C.\left(-J^{-1}\frac{dR}{d\Delta t},1\right)^T,$$

$$\hat{\delta}\leftarrow\frac{\delta}{\|\delta\|}. \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (10)$$

This can always be done because C is positive, and so the tangent norm is bounded away from zero. This normalization is valid even when the solution is at a steady state. Eq. 10 provides a computationally attractive way to compute level curve tangents at points near the solution path. At such points, the Jacobian is numerically well conditioned because it is evaluated in the vicinity of a unique solution and it has the same structure as standard reservoir simulation Jacobian matrices. Algorithm 2 in Appendix B prescribes the details of computing the tangent vector at a point (U, Δt) close to the solution path emanating from the point (U^n, 0).

Defining a Convergence Neighborhood

A key component of the CN algorithm is having a quantitative measure of proximity to the solution path in the augmented space. A tight measure results in iterates that are accurate solutions but may result in poor computational efficiency because the solution path in the (U, Δt) space is needlessly traced in detail. On the other hand, a loose tolerance may produce iterates that are farther away from the solution, requiring more Newton corrective steps. Accordingly, one objective is to select this proximity measure so that points within the neighborhood around the (U, Δt) solution path provide good starting points for a Newton corrective process for their corresponding timestep. A second objective is to select this neighborhood so that there is a computationally inexpensive procedure to test whether a given point is within it or not.

We denote the solution path emanating from $p_0=(U_0,0)\in\mathbb{R}^{N+1}$ as the set of points $\mathcal{C}=\{p(\lambda)=[U(\lambda),\Delta t(\lambda)]: R[U(\lambda),\Delta t(\lambda);U_0]=0,\lambda\geq 0\}$.

Three measures and their corresponding convergence neighborhoods, $\mathcal{N}$, maybe defined as follows:

- Residual or material-balance norm. A point is in the neighborhood provided that its residual norm, or material-balance error, is less than a specified tolerance; $p\in\mathcal{N}$ if and only if $\|R(p)\|\leq\varepsilon_{\text{tol}}$.
- Absolute or maximum change tolerance estimates. At any point (U, Δt), the norm of the first Newton step, $\left\|-J^{-1}R\left(U,\Delta t;U^n\right)\right\|$, toward the solution path can be related to the norm of the absolute error. In particular, it is a linearized estimate that becomes more accurate as a measure of absolute error within the vicinity of the solution path.
- Jacobian-matrix norm or curvature estimates. Within the convergence basin of the Newton flow, the Jacobian matrix induces a Lipschitz property on the residual. That is, within the neighborhood of the solution, the curvature of the residual is bounded. Using this property, a computational estimate of a Kantorovich condition on the Jacobian-matrix norm can be used to estimate whether the point in question is a good point to start a Newton iteration [see, for example, Deuflhard (2004) and Ortega and Rheinboldt (1970)].

Residual-based measures require the computation of the residual only, and they are accurate measures of proximity if one is close to the solution. Residual measures alone, however, do not necessarily guarantee favorable convergence rate properties. Jacobian matrix norm measures invariably require computing the Jacobian and provide estimates of the local convergence properties of a Newton iteration. They do not measure proximity directly, however. Absolute-error estimates are the most computationally expensive, requiring the computation of a Newton step. Absolute-error measures can combine more-accurate measures of both proximity to the solution and convergence-rate properties. A combination of

these three broad types of measures may be used to ensure robustness while trading off computational performance. Note that the strictest criterion is to use a residual tolerance so that every point in the neighborhood of the augmented solution path is itself an acceptable solution for the residual equations. While such a strict criterion will always generate actual solution iterates, it may lead to needlessly following the solution path too closely, and that can increase the number of continuation steps per target timestep. In this work, we apply a residual tolerance criterion only. Algorithm 3 in Appendix B describes the details of testing whether a given point (U, Δt) is within the convergence neighborhood about the solution path emanating from (U_0, 0).

Step Length Selection Along Tangents

Another component of the CN algorithm is the selection of an appropriate step length along a tangent vector. The CN update is written as

$$\begin{pmatrix} U \\ \Delta t \end{pmatrix}_{k+1} \leftarrow \begin{pmatrix} U \\ \Delta t \end{pmatrix}_{k} + \alpha \begin{pmatrix} \hat{\delta}_U \\ \hat{\delta}_{\Delta t} \end{pmatrix}, \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (11)$$

where α is the positive step length.

A larger step length results in a larger timestep advancement because the timestep component of the tangent, $\hat{\delta}_{\Delta t}$, is always positive. This implies a higher computational efficiency as the timestep advancement per tangent computation increases. On the other hand, a larger step length also implies a larger absolute error from the solution path. Using continuity around solutions, theory guarantees that, for small step lengths from points on the solution curve, any residual tolerance may be satisfied. In practice, CN iterates are not exactly on the solution path, however, and, for a given tangent, there may be no positive step length that results in an acceptable point that lies within the convergence neighborhood, $p_{k+1} \in \mathcal{N}$. In such cases, the step length selection algorithm must trigger Newton correction steps. This switch to Newton corrections is dictated by the nature of the physics being modeled. For example, slowly transient solutions can be stepped through with larger tangent step lengths, while faster transients may require shorter tangent steps.

In practice, a minimal step length, $\alpha_{\min}$, is chosen as a parameter. Note that, from Eq. 11, for a given step length α, the corresponding updated timestep size is simply $\Delta t^k + \alpha \hat{\delta}_{\Delta t}$. We can choose $\alpha_{\min}$ to satisfy a minimal timestep advancement per tangent step, $\Delta t_{\min}$.

There may be many step lengths that are larger than the minimum, $\alpha > \alpha_{\min}$, that keep the new iterate within the convergence neighborhood. In this work, the objective of the step length algorithm is to select the largest such step length in order to maximize the timestep advancement achieved per continuation step. Because constrained univariate optimization is generally more efficient than an unconstrained counterpart, we also set a maximal step length parameter, $\alpha_{\max}$. Choices for the maximal step length parameter, $\alpha_{\max}$, can be made in several ways. One approach could be to choose the smallest step length that makes all updated normalized variables reach their physical range. Another approach, which is used throughout the examples in this work, is to select it so that the tangent timestep update satisfies a maximal timestep advancement per tangent step, $\Delta t_{\max}$.

The univariate optimization problem for the tangent step length can be expressed as

$$\begin{cases} \text{maximize } \alpha \\ \text{subject to } p_k + \alpha \hat{\delta}^k \in \mathcal{N}, \\ \qquad \alpha \in \left[\alpha_{\min}, \alpha_{\max}\right] \end{cases} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (12)$$

In the examples in this work, we apply a derivative-free backtracking algorithm to solve this problem. The details are presented in Algorithm 4 in Appendix B.

Illustrative Examples Using CN

We consider two illustrative cases of Problem 1, which is described in Appendix A. The first case is a horizontal piston-like displacement, and the second includes gravity effects and exhibits countercurrent flow. In both cases, the injection saturation is unity, $S_{\text{inj}} = 1$, and the relative permeability functions are quadratic with endpoints of zero and one.

Horizontal Buckley-Leverett Displacement. For this case, the endpoint mobility ratio M^0 is chosen as 10; the gravity number Ng is chosen as 0 for a horizontal problem; the initial saturation, S_{init}, is zero; and $N = 150$.

We illustrate the CN concepts by applying a step of the algorithm for this problem, starting from a simulation time of 0.5. At this time, the saturation state, denoted S^0, and the tangent update $\hat{\delta}^0$ computed using Algorithm 2, are presented in **Fig. 4.**

The CN starting iterate to solve a target timestep from this time consists of the saturation profile, S^0, in Fig. 4, augmented with a timestep size of zero; that is, $p_0 = (S^0, 0)$. The solution path emanates from p_0 and is in an N+1-dimensional space. The tangent to the solution path at the starting point is the state update component plotted in Fig. 4, augmented with $\frac{d\Delta t}{d\lambda}$, which in this case is 0.336. The next CN iterate is a step from the starting point along a linearly scaled update of the tangent. This update is written as $p_1 \leftarrow p_0 + \alpha \hat{\delta}$, where the scalar step length, $\alpha_{\min} \le \alpha \le \alpha_{\max}$, is to be selected by Algorithm 4.

We illustrate the qualitative nature of this tangent update using two different step lengths, $\alpha = 0.015$ and 0.089. For each step length, the corresponding tangent-update timestep size is $\alpha \frac{d\Delta t}{d\lambda}$, and, in this case, these are 0.005 and 0.03. Moreover, defining the Courant-Friedrichs-Lewy (CFL) number as $\frac{\Delta t}{\Delta x} \max |f'(S)|$, these step lengths correspond to 2.25 and 13.35 CFL. **Figs. 5a and 5b** show the initial saturation state and the updated states obtained by taking a single tangent step using each of the two selected step lengths. The solutions for each of the two timestep sizes, which correspond to the two step lengths, are also shown in Figs. 5a and 5b. In these figures, the solutions are obtained by a Newton process, which requires several iterations, whereas the tangent steps are single iterates in the CN algorithm. Qualitatively, it is observed that, as the step length is increased, the tangent step becomes a worse approximation of its corresponding solution, as expected. The objective is to choose the largest step length that still gives a good approximation (i.e., is close to the solution path).

Fig. 6 shows the trend in the residual norm for various choices of step length along the tangent. The figure also shows the residual norms obtained using the old state as an initial guess for the corresponding timestep size of the continuation step length. Within smaller step length ranges, there is higher confidence in the tangent update solution than in the old state. The step-length selection

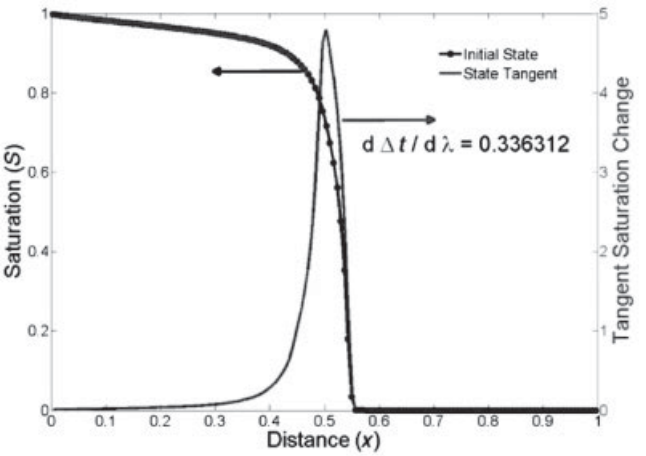


Fig. 4—Starting iterate and tangent update for a 1D Buckley-Leverett problem with no gravity.

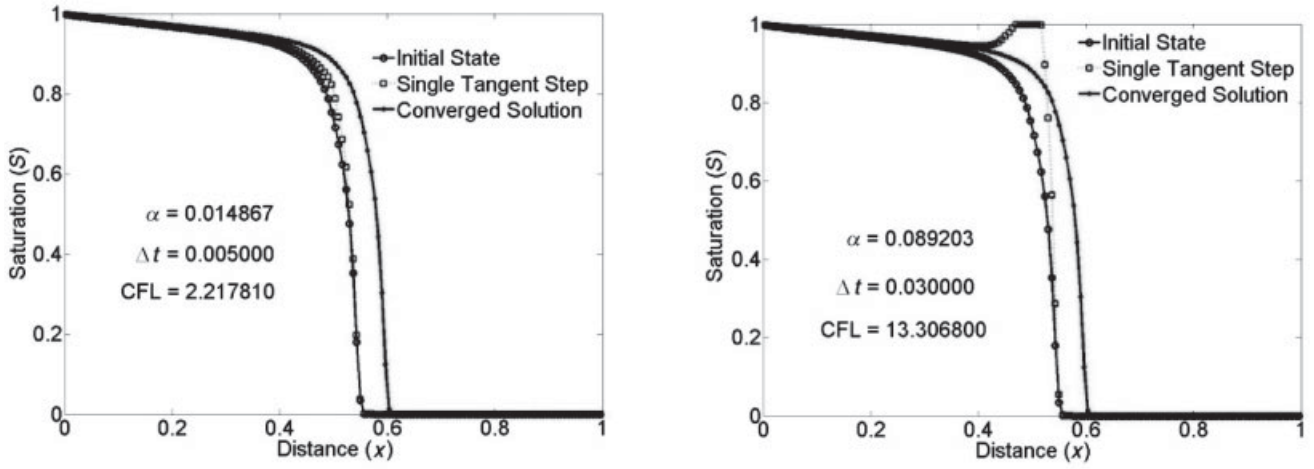


Fig. 5—For a particular time, the current state is depicted with circle markers, the tangent update with star markers, and the actual solution with square markers. This is presented for two different continuation step lengths.

algorithm attempts to select the largest step length that retains a residual below a certain tolerance. For low residual tolerances, which are typically used to define a convergence neighborhood, the continuation tangent step affords a timestep advancement that is (1) known a priori through the relation $\Delta t^1 \leftarrow \Delta t^0 + \alpha \frac{d\Delta t}{d\lambda}$, and (2) always closer to the solution than the old state is.

In terms of the overall performance of CN for a single timestep, **Fig. 7a** shows the number of iterations required to solve a set of different target timestep sizes ranging up to a size corresponding to more than 440 CFL. For each target timestep size, starting from the simulation time 0, the iterations are performed using the (1) Eclipse-Appleyard-Newton approach, (2) modified-Appleyard-Newton method, and (3) the proposed CN algorithm. In Fig. 7a, we show the split between the number of iterations taken in CN tangent steps and in CN Newton corrections.

The EA Newton algorithm does not converge for any of the timestep sizes attempted. Moreover, while in general the CN and MA approaches are expected to perform on par, in this case the CN algorithm requires fewer overall iterations. This is the case because the additional dimension of the tangent update, $\frac{d\Delta t}{d\lambda}$, provides a temporal scaling regarding the range of validity of an update, whereas scaling an MA Newton step never influences the timestep that it attempts to solve. Fig. 7b shows the number of residual evaluations required to converge a timestep by each of the methods. Owing to the step length search component of the CN algorithm, it requires more residual evaluations. The precise number required is bounded by the maximum allowed backtracking steps per tangent step. In this case, they were limited to five. Note that, although in this case the proposed algorithm performs on par with a state-of-the-art solver, each of the CN iterates is associated with a timestep size. Subsequently, in return for a few additional evaluations of the residual, the CN algorithm does not require timestep chops, nor will it waste any iterations over the course of an entire simulation.

Down-Dip Buckley-Leverett Displacement. For this case, the endpoint mobility ratio, M^0, is chosen as 0.5, and the gravity number, Ng, is chosen as −5 for a down-dip problem. The number of gridblocks used is $N = 150$. The initial condition has a saturation of 1 up to a distance of 0.34 and a saturation of 0 elsewhere.

We consider a single timestep starting at the time of 0.1, and we illustrate the components of a single CN iteration. The current saturation state and the computed tangent update are shown in **Fig. 8.** In this case, countercurrent flow of the two phases results

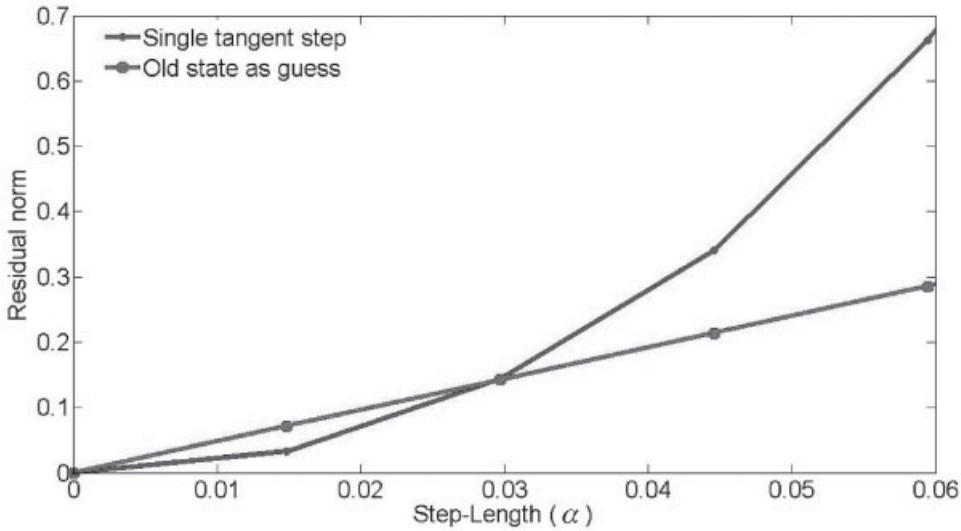
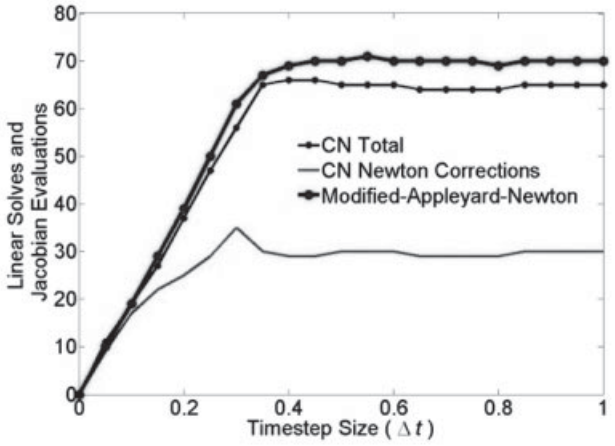
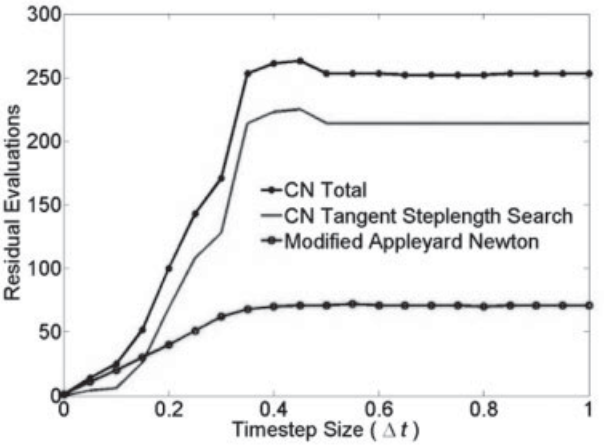


Fig. 6—Residual norms for various step lengths along a single tangent.



(a) Number of Jacobian evaluations and linear solves required for convergence using CN and modified-Appleyard-Newton methods.



(b) Number of residual evaluations required for convergence using CN and modified-Appleyard-Newton methods.

Fig. 7—Convergence characteristics of the CN method compared to the modified Appleyard-Newton method.

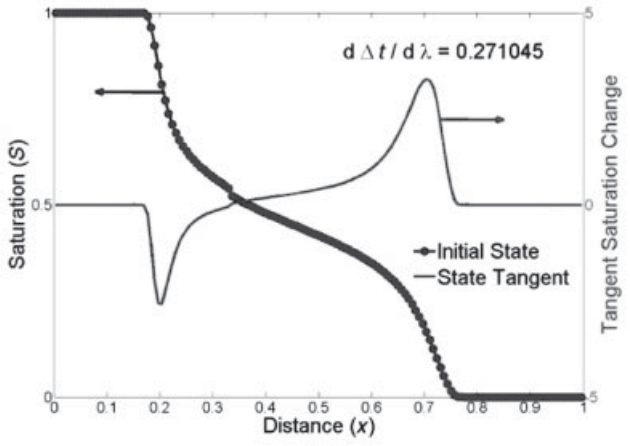
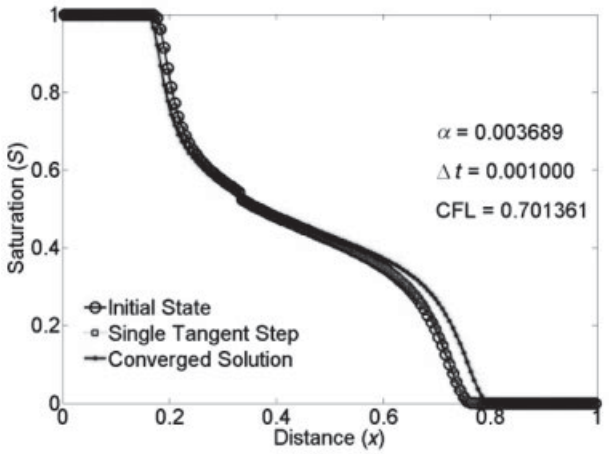


Fig. 8—A starting point for a continuation timestep and the corresponding initial tangent vector.

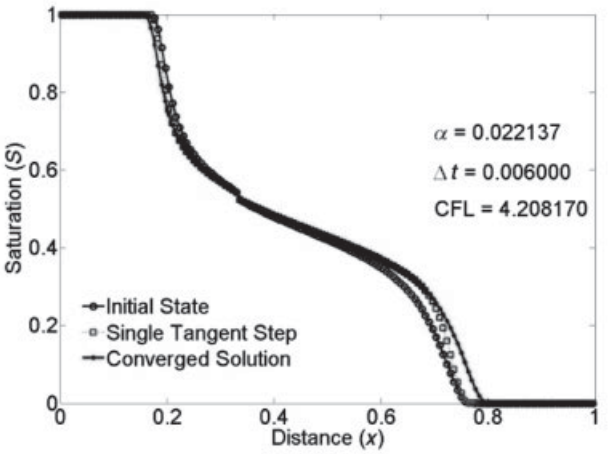
in negative components of the tangent vector. Moreover, the displacement involves spreading wave portions. For two step lengths corresponding to timestep sizes of 0.001 and 0.006, **Figs. 9a and 9b** show the initial state, the tangent iterate, and the corresponding complete solution for the timestep. Defining the CFL number for this problem as $\frac{\Delta t}{\Delta x}\max|f'(S)|$, the timesteps correspond to 0.7 and 4.2 CFL. Qualitatively, the linear tangent updates with large step lengths are worse approximations in the sense that they display sharp changes in saturation.

Fig. 10 shows the relationship between the computed residual norm and increasing step length. Once again, for smaller residual tolerances, the tangent approximations provide better estimates than the old state.

Figs. 11a and 11b show the overall performance for various timesteps from the start of the simulation. The largest timestep size tested is 0.2, which corresponds to a CFL number of 140. In this case, the EA Newton method converges only for the smallest timestep size. Moreover, beyond a timestep size of 0.04, corresponding to a CFL number of 30, the CN algorithm requires more iterations than the MA Newton method. This is because the solution involves a spreading wave that influences the entire domain.



(a)



(b)

Fig. 9—For a particular time, the current state is depicted with a solid line, the tangent update with a dotted line, and the actual solution with a dashed line. This is presented for two different continuation step lengths.

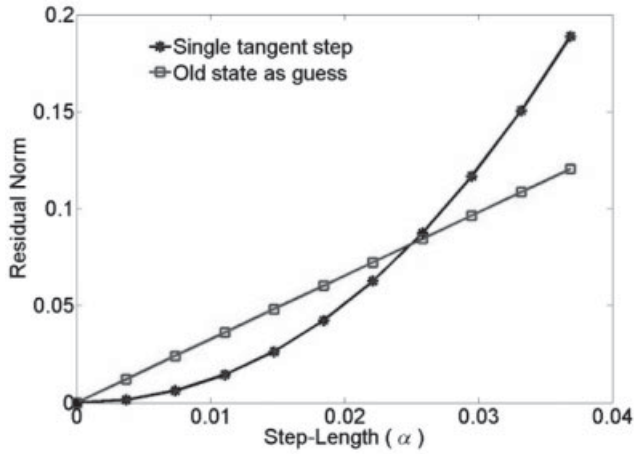


Fig. 10—The residual norm vs. step length size using a tangent step, and the initial state as a guess.

Summary

The CN algorithm evolves the residual equations in the augmented (U, Δt) solution space, providing iterates that are solutions to known timestep sizes. Subsequently, whenever a CN iteration is terminated, we obtain either the solution to the target timestep or a solution to a smaller known timestep. The rate of convergence is empirically shown to be on par with that of state-of-the-art safeguarded Newton methods whenever such schemes converge. Qualitatively, the number of iterations required for convergence by both the variants of Newton's method and the CN scale with the dimensionless target timestep. This is because both iterations exploit linearized local wave propagation speed information. The chief difference is that, while Newton's method is oblivious to the nature of the time evolution of the solution within a timestep, the CN algorithm exploits the evolution of the discrete residual equations in time.

The next section introduces a computational method to reduce the proportionality between the computational effort required for convergence and the timestep size.

Performing Computation Only Where It Is Needed—Adaptive Localization

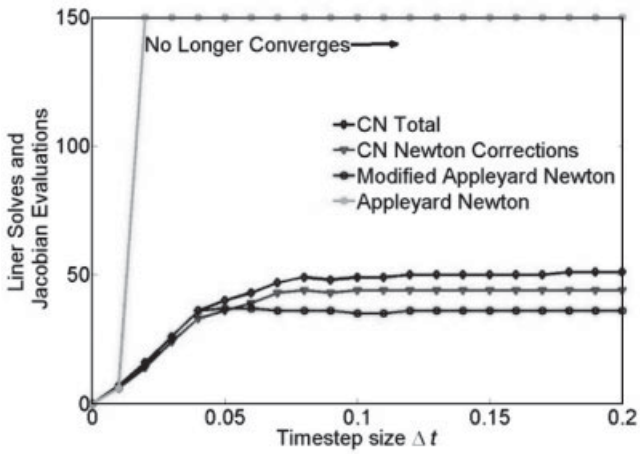
With CN, we guarantee that, at the end of an iteration, we have either the solution to the target timestep or a solution to another known yet smaller timestep. Here, we devise a method to reduce the amount of effort required to compute each iterate during the CN process. This is desirable because, in addition to the convergence guarantees, we want to improve the computational efficiency by obtaining a higher ratio of timestep advancement to computational effort.

Current approaches for reducing the amount of computation without ignoring physical coupling are centered around substituting the original large problem with a smaller one whose solution, in some sense, is representative of the original one (Chen and Durlofsky 2006; Muller and Stribia 2007; Plewa et al. 2003). In particular, these methods attempt to exploit the property of local domain of dependence, which is characteristic of hyperbolic conservation laws in order to choose a coarse representation to be solved. In such approaches, it may be challenging to form a coarse model that results in a solution that is representative of the original nonlinear problem.

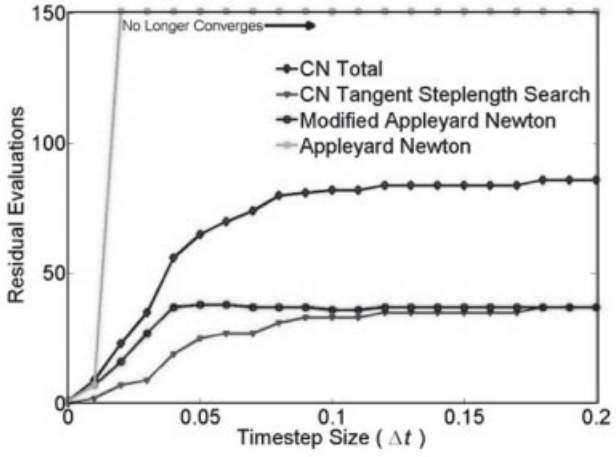
Another approach toward developing adaptive methods is to stick to a standard scheme and spatial mesh and to then build adaptivity into the underlying solver. Because neither the mesh nor the scheme is modified, there will be no introduction of accuracy or stability issues. Large-scale variants of Newton's method, for example, reduce the computational effort required for each iteration without degrading the local nonlinear convergence rate (Keyes 2002; Watson 1986). Common approaches to achieving this are inexact-Newton (IN) and quasi-Newton (QN) methods, or a combination of both (Deuflhard 2004; Ortega and Rheinboldt 1970).

IN methods apply approximate linear solvers to compute the Newton direction, given a precise Jacobian matrix. When combined with parameterized controllers for the linear-solver accuracy, IN methods can effectively manage an efficiency-to-accuracy trade-off. For example, Krylov-Newton controllers can compute crude Newton steps away from the solution and more-accurate ones within the quadratic convergence basin. QN methods, on the other hand, use approximations to the Jacobian matrix itself, thereby reducing the computational effort required to compute it. While it is challenging to control the accuracy of such updates, QN methods, such as the Broyden update class, are the preferred method when the Jacobian itself is not available for computation (Broyden 1965; Broyden et al. 1973).

Our objective is to complement IN and QN strategies with the ability to considerably reduce the dimensionality of the linearized system while still guaranteeing accurate computation of Newton directions. We accomplish this by specializing to conservation equations with traveling waves. Specifically, we exploit the nature of the finite domain of dependence in order to identify the components that will change the most over a Newton step. Then, only these specific components need to be computed by a linear solution strategy.



(a) Number of Jacobian evaluations and linear solves required for convergence using CN and modified-Appleyard-Newton methods.



(b) Number of residual evaluations required for convergence using CN and modified-Appleyard-Newton methods.

Fig. 11—Convergence characteristics of the CN method compared with modified Appleyard-Newton method.

We achieve this by following nonzero entries in the right-hand side of the tangent solution, $\frac{\partial R}{\partial \Delta t}$, one at a time through the directed flow graph of the Jacobian matrix. We show that upwind schemes have the property that the sources do not propagate very far. In fact, the influence decays geometrically according to a derived relation in the local cell-face linearized wave speeds.

Empirically, pressure unknowns in near-elliptic problems show a similar property. That is, the pressure changes are greatest around saturation changes, and these pressure changes decay outward, though with an extended support. No well-established theory is currently available to quantify this support of the pressure field, however, as we have for the other unknowns (saturation and composition). We develop and demonstrate our algorithmic framework starting from scalar hyperbolic equations and moving onto hyperbolic systems and systems of coupled parabolic/hyperbolic equations.

High-Level View of Adaptive Localization

Reservoir simulation problems often involve concentration waves that are local to certain portions of the domain. The upwind directions on a simulation mesh form a network through which mass transfer takes place over the duration of a timestep. The continuation tangent vectors in Figs. 4 and 8 are examples of linearized state updates that propagate material-balance errors over the domain during a timestep. These updates are computed using the inverse of the simulation Jacobian matrix, which itself incorporates upwinding information as well as local wave speeds. Such updates are typically nonzero only within the vicinity of sharp fronts in the old state. The objective is to devise a quantitative and reliable strategy to exploit this locality before solving the linear system.

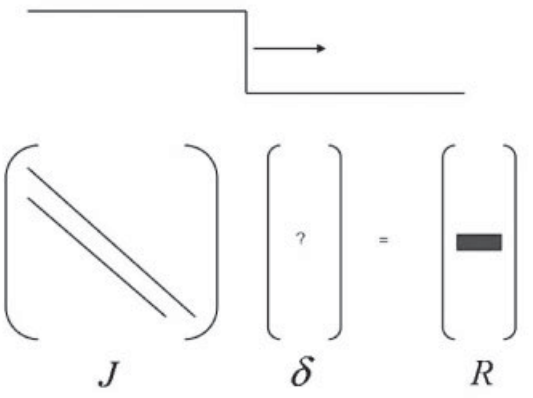
Fig. 12a is a depiction of the problem of computing a linear update, δ, to an initial saturation state for a given timestep. The saturation is assumed to be a piston-like front, and the flow is from left to right. Consequently, the residual, R, has a single nonzero entry, isolated to the cell flanking the front. The Jacobian matrix, J, is lower bidiagonal. Fig. 12b shows the action of the inverse of the Jacobian on the residual. The inverse of the Jacobian is lower triangular, and it can be shown that its entries decrease in magnitude away from the main diagonal. Consequently, the update is nonzero only in the cell flanking the front and in the cells downstream of it. The magnitude of the update can be shown to be largest in the frontal cell and decreases rapidly along the downstream direction. This observation motivates an approach that isolates nonzero entries in the right-hand side and independently inspects their propagation throughout the directed graph of the Jacobian matrix. Using the principle of superposition on these isolated nonzero entries, we can obtain the solution to problems with general right-hand-side vectors. Using this technique, it is possible to identify which cells will experience linear updates larger than a prescribed magnitude before solving the system. As a result, only a smaller subsystem corresponding to these isolated cells needs be solved. In the hypothetical example of Fig. 12b, this reduced subsystem is indicated by the dotted box.

General Localization Algorithm

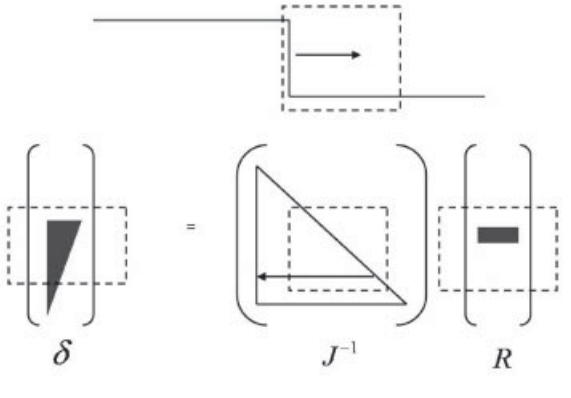
A Newton step and a CN tangent are computed by solving a linear system involving the Jacobian matrix with different right-hand-side vectors. Whereas Newton steps have the residual vector as a right-hand side of the system, CN tangent steps have the vector term $\frac{\partial R}{\partial \Delta t}$ on the right-hand side. Without loss of generality, we will consider the case of solving for a Newton step, $\delta = J^{-1}R$.

By linearity, the Newton step can be written as the sum of N substeps, $\delta = \delta_1 + \ldots + \delta_N = J^{-1}R_1 + \ldots + J^{-1}R_N$, where each substep is the Newton component that solves a material-balance error in a single cell. That is, the residual component, R_j, is a vector with only one logically nonzero entry occurring in the jth component. As already indicated, the residual vector of multiphase-flow problems is typically sparse. Subsequently, several residual components may be zero. The localization algorithm exploits this principle of superposition in order to estimate or directly obtain the Newton substeps corresponding to the nonzero residual components.

Owing to the local nature of saturation and concentration waves, the individual Newton substeps can be obtained or estimated without solving the entire system. This is because each substep has a right-hand side with one nonzero entry. A theoretical development of this process is presented in Appendix C for hyperbolic conservation laws in 1D. The solution process for such substeps extends to multidimensional problems on general meshes. **Fig. 13** illustrates this. The application of the principle of superposition, where we treat nonzero residual entries independently, holds. The entries in the Jacobian matrix form a weighted directed graph through the mesh. By generalizing the 1D relations derived in Appendix C, we can apply the same procedure to identify cells with nonzero updates. In particular, for problems where there are no directed cycles within the upwind graph, this is achieved by first performing a breadth-first traversal originating at the cell with the nonzero material-balance error. For an introduction to the breadth-first-traversal graph algorithm, see Cormen et al. (2001). By a second traversal limited to the cells that were already visited, the update components are computed sequentially. This is the case for all scalar problems in saturation and for multiphase problems without countercurrent flow (Kwok and Tchelepi 2007).



(a) Schematic of the solution for a linearized update to a piston-like displacement over a timestep.



(b) The linearized update is local to the front and decays along the upwind direction.

Fig. 12—An illustration demonstrating the locality of computed linear updates (continuation tangents or Newton steps) for a 1D piston-like front.

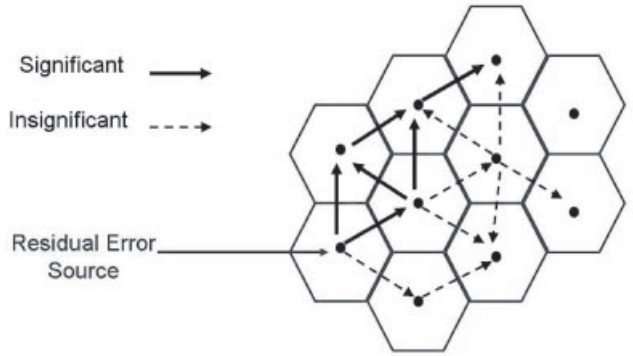


Fig. 13—A single nonzero entry in the residual can be tracked through the directed upwind graph to determine its range of influence on the corrective linearized update.

If the upwind graph, on the other hand, has directed cycles, heuristics must be employed to estimate the range of influence. This is the case for problems with countercurrent flow and with compressibility. In such cases, the heuristic we employ is to disconnect edges that cause cycles. The effect of this is not expected to be severe because the subsequent iteration can be shown to spread the material-balance error onto a larger proportion of the domain, requiring a complete system solve because of these global interactions. For a detailed description of this algorithm, see Younis (2009).

Performance of the Adaptively Localized CN (ALCN) Algorithm

A suite of cases of Problem 2, which is described in Appendix A, is used to investigate the properties of the proposed ALCN algorithm. We present results for two representative cases. The first is a gravity-segregation problem with a nonstationary initial condition. This case is known to stress the nonlinear-solution aspect of the transport. The second case superposes gravity effects and nonlinear relative permeability with injection and production wells. Both cases share the following common parameters. The square reservoir is assumed to have no-flow boundaries, with dimensions of 100 ft, and is discretized using 100 blocks along both dimensions. An initial pressure of 500 psi is applied to the top of the reservoir. We assume a uniform initial porosity of $\phi_{ref} = 0.4$ at a reference pressure of $P_{ref} = 14.7$ psi, a rock compressibility of $c_r = 1\times10^{-6}$ psi^{-1}, and an isotropic permeability field of 500 md. The normalized relative permeability models used are quadratic and are scaled to endpoints of 0 and 1. The oil and water have gravimetric densities of 50 and 100, and viscosities of 5 and 1cp, respectively. The residual 2-norm convergence tolerance of 1×10^{-6} is applied, and a CN convergence neighborhood tolerance of 1×10^{-3} is used.

Pure Gravity Segregation

We consider a case of pure gravity segregation where, initially, oil saturates the lower portion of a reservoir and water saturates the top. Because the boundary conditions are no-flow, the problem involves bouncing waves that interact every time they collide with one another or with the top and bottom reservoir boundaries. The steady-state solution is an equilibrium with oil on the top. **Figs. 14a through 14c** show oil-saturation snapshots over the course of a simulation.

In order to derive an appropriate cell-based CFL number, we assume incompressible flow and a zero total velocity. Given these assumptions, the cell-based CFL number for cell i, is given as

$$CFL_i = \Delta t \frac{K(\gamma_w - \gamma_o)}{\phi_i \mu_o \Delta y} \max F'(S), \quad \ldots\ldots\ldots\ldots\ldots\ldots (13)$$

where $F'(S_i)$ is the slope of the following flux function:

$$F(S_i) = \frac{K_{ro}}{1 + \frac{K_{ro}}{K_{rw}} \frac{\mu_w}{\mu_o}}. \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (14)$$

Fig. 15a shows the sorted distribution of CFL numbers throughout the mesh. These cell-based CFL numbers are computed using the initial condition. For timestep sizes greater than 4.5 days, all of the cells in the domain experience CFL numbers that are greater than unity.

Fig. 15b shows the iteration count to convergence during a series of experiments. In each experiment, a target timestep is set from the initial condition and the corresponding system is solved. A maximum number of 150 iterations is allowed, and an iteration count of 150 implies lack of convergence. The SN and EA methods failed to converge for the smallest timestep size tested. Fig. 15b shows that the proposed algorithm provides improved asymptotic convergence rates, as well as robustness compared with the MA method. **Fig. 16** shows the average fraction of the total unknowns that is solved for at each iteration using ALCN. The fractions vary with timestep size because the locality of the saturation changes also varies in time. As the segregation process approaches steady state, we observe greater locality, whereas, at early times, the saturation over the entire domain experiences a slowly varying spreading wave. Moreover, because the Newton correction steps within the ALCN algorithm are computed within the convergence neighborhood, we see that they modify only a small percentage of the unknowns.

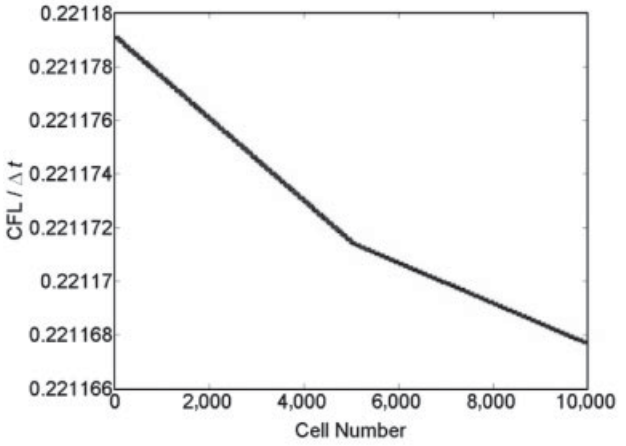
Gravity Effects Coupled to Well-Driven Flow

Next, we consider a case that involves both gravity segregation and well-driven flow. Initially, the reservoir is saturated with oil in the upper half and with water in the lower half. A single block injector is completed in the upper-right corner, and a producer is completed in the lower-left corner. A constant water-injection rate of 10 B/D is specified. The producer is operated with a constant bottomhole pressure of 500 psi. **Figs. 17a through 17c** show water-saturation snapshots at 0, 120, and 245 days during the course of a simulation.

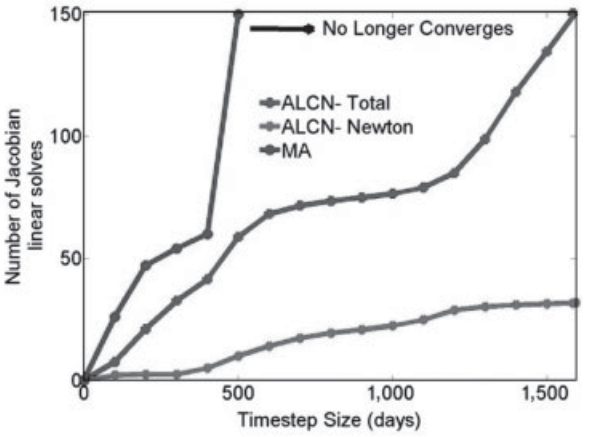
The results in **Fig. 18** illustrate the computational effort required to solve a full simulation using a single timestep. Various timestep

(a) 0 days (b) 1,350 days (c) 4,500 days

Fig. 14—Oil-saturation snapshots during simulations for a gravity-segregation case.



(a) Sorted distribution of CFL numbers throughout the domain.



(b) The number of iterations required to solve a set of sample timestep sizes using the modified Appleyard heuristic and the proposed ALCN algorithm.

Fig. 15—Results for a gravity-segregation problem with compressibility.

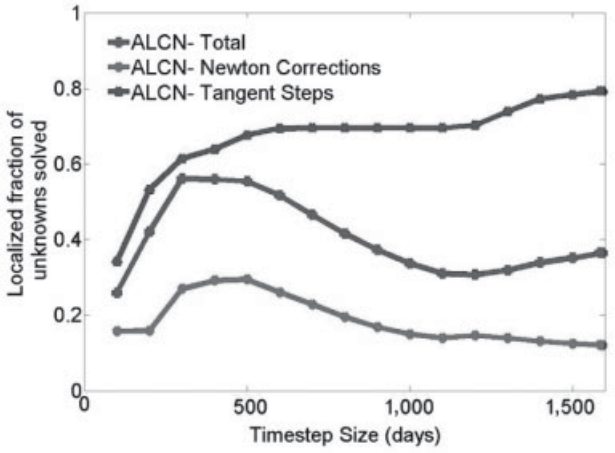


Fig. 16—The average fraction of unknowns solved for at each iteration using localization.

sizes are tested using the proposed ALCN algorithm, as well as the MA method. A dimensionless measure of these timestep sizes is the cell pore volumes injected (CPVI), which, for this case, corresponds to 10 CPVI for a timestep size of 1 day. Fig. 18a shows the count of linear solves required to converge a timestep using both methods. Once again, the proposed ALCN algorithm provides improved asymptotic convergence rates over the tuned MA method. Moreover, both the SN and EA methods fail to converge for the smallest timestep size tested. Fig. 18b shows the count of residual evaluations required by the solution processes. Note that the ALCN method does require more residual evaluations because of the step length selection procedures within the ALCN method.

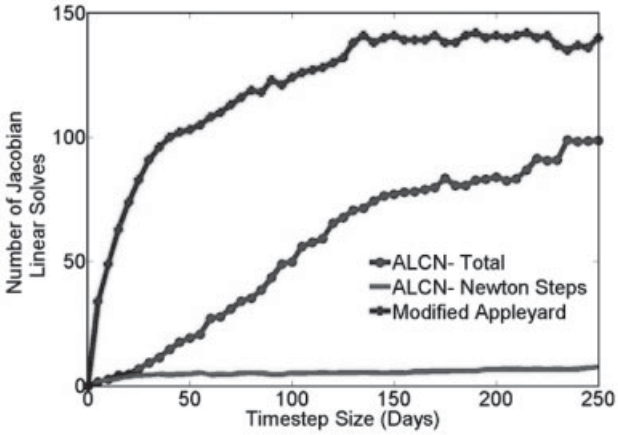
Concluding Remarks

We developed and illustrated an algorithm that solves implicit residual systems using a combination of Newton's method and a continuation on timestep size. The algorithm guarantees convergence for any timestep size, and, if the iteration process is stopped before the target timestep is reached, the last iterate is a solution to a smaller known timestep. The performance of the algorithm was illustrated using several challenging nonlinear problems. Compared to state-of-the-art problem-tuned heuristic methods, the CN algorithm remains competitive. Additionally, the CN algorithm does not require wasteful timestep cuts, unlike all Newton variants. Moreover, a localization strategy that is based on solution propagation properties of the transport equations is used to reduce the computational cost associated with solving the linear systems. With the combination of these key ideas, reservoir simulators no longer require timestep-chopping heuristics, no computational effort can be wasted, and the amount of computation required per iteration can be reduced substantially.

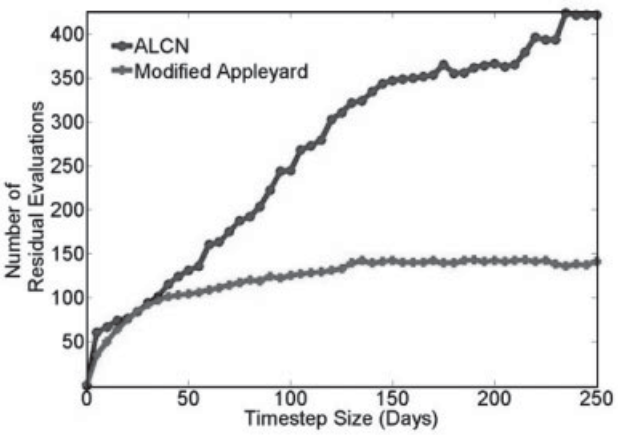
The CN algorithm presented in this work applies a first-order explicit scheme to follow the solution path loosely. We anticipate increased computational efficiency through the development of stabilized CN variants, which exploit the relationship between Newton and tangent steps within the augmented space. This may

(a) 0 days (b) 120 days (c) 245 days

Fig. 17—Water-saturation snapshots for an unstable injection problem with gravity.



(a) The number of linear solves (iterations) required to solve a set of sample timestep sizes using the modified Appleyard heuristic and the proposed ALCN algorithm.



(b) The number of residual evaluations performed during the solution of a set of sample timestep sizes using the modified Appleyard heuristic and the proposed ALCN algorithm.

Fig. 18—Computational effort required to solve a full simulation in one timestep using the proposed ALCN algorithm and the modified Appleyard-Newton method.

lead to a characterization of the CN convergence rate in terms of the order of accuracy in approximating the solution path.

We also anticipate the application of CN in order to introduce timestep control on the basis of accuracy concerns. That is, the CN algorithm offers a built-in facility to perform timestep selection for the control of time-approximation errors. Because each iterate solves a larger timestep than its successor, error-extrapolation techniques may be applied directly to dictate which timestep to stop at, thereby providing an a priori error controller that is built into the solver and that does not waste computation.

Nomenclature

$\mathcal{C}$ = solution path in augmented space
C = a positive constant
i = unknown index
J = Jacobian matrix
j = index of cell with nonzero material balance
k = CN iteration index
M^0 = endpoint mobility ratio
$\mathcal{N}$ = convergence neighborhood about a solution path in augmented space
Ng = gravity number
n = timestep number
p_k = point in the augmented space
p_{int} = point in the augmented space that is within the interior of a convergence neighborhood
R = residual system of nonlinear equations
S = saturation unknowns
S_{inj} = injection saturation
S_{init} = initial saturation
S_{int} = saturation state within the interior of a convergence neighborhood
t_n = time at timestep number n
U^n = vector of unknowns at timestep number n
U_0 = vector of unknowns at the beginning of a CN process
α = CN tangent update step length
$\alpha_{\max}$ = maximum allowable CN tangent update step length
$\alpha_{\min}$ = minimum allowable CN tangent update step length
$\Delta\nu$ = step length along a Newton direction
Δt = timestep size
Δt_{target} = target timestep size
Δt_{int} = timestep component of a point within a convergence neighborhood in augmented space
δ = CN tangent in augmented space
$\hat{\delta}$ = normalized CN tangent in augmented space
δ_U = state update component of CN tangent
$\delta_{\Delta t}$ = timestep update component of CN tangent
ζ = local scaling constant
θ = local attenuation ratio
Λ = diagonal weighting matrix for a general safeguarded Newton iteration
λ = solution path arc-length parameterization
ν = Newton iteration index

Acknowledgments

This paper was prepared with support from the Stanford University Petroleum Research Institute-B (SUPRI-B) Industrial Affiliates Program. The authors thank Garf Bowen and Larry Fung for their constructive discussions. The authors also thank Garf Bowen and John Appleyard for sharing insights regarding the Eclipse Appleyard heuristic.

References

Allgower, E.L. and Georg, K. 1980. Simplicial and continuation methods for approximations, fixed points and solutions to systems of equations. *SIAM Review* **22** (January): 28–85.

Allgower, E.L. and Georg, K. 2003. *Introduction to Numerical Continuation Methods*, No. 45. Philadelphia, Pennsylvania: Classics in Applied Mathematics, Society for Industrial and Applied Mathematics (SIAM).

Aziz, K. and Settari, A. 1979. *Petroleum Reservoir Simulation*. Essex, UK: Elsevier Applied Science Publishers.

Broyden, C.G. 1965. A class of methods for solving nonlinear simultaneous equations. *Mathematics of Computation* **19** (92): 577–593. doi: 10.2307/2003941.

Broyden, C.G., Dennis, J.E., Jr., and Moré, J.J. 1973. On the Local and Superlinear Convergence of Quasi-Newton Methods. *IMA Journal of Applied Mathematics* **12** (3): 223–245. doi: 10.1093/imamat/12.3.223.

Chen, Y. and Durlofsky, L. 2006. Adaptive Local-Global Upscaling for General Flow Scenarios in Heterogeneous Formations. *Transport in Porous Media* **62** (2):157–185. doi: 10.1007/s11242-005-0619-7.

Cormen, T., Leiserson, C., Rivest, R., and Stein, C. 2001. *Introduction to Algorithms*, second edition. Cambridge, Massachusetts: MIT Press.

Deuflhard, P. 2004. *Newton Methods for Nonlinear Problems: Affine Invariance and Adaptive Algorithms*, No. 35. Berlin: Series in Computational Mathematics, Springer-Verlag.

ECLIPSE Technical Description 2008.2. 2008. Houston: Schlumberger GeoQuest.

Gropp, W.D., Kaushik, D.K., Keyes, D.E., and Smith, B.F. 2001. High performance parallel implicit CFD. *Parallel Computing* **27** (4): 337–362. doi: 10.1016/S0167-8191(00)00075-2.

Jenny, P., Tchelepi, H.A., and Lee, S.H. 2009. Unconditionally convergent nonlinear solver for hyperbolic conservation laws with S-shaped flux functions. *Journal of Computational Physics* **228** (20): 7497–7512. doi: 10.1016/j.jcp.2009.06.032.

Keyes, D.E. 2002. Terascale implicit methods for partial differential equations. In *Contemporary Mathematics 306: Recent Advances in Numerical Methods for Partial Differential Equations and Appliations*, ed. X. Feng and T.P. Schulze, 29–84. Providence, Rhode Island: American Mathematical Society (AMS).

Kwok, F. and Tchelepi, H. 2007. Potential-based reduced Newton algorithm for nonlinear multiphase flow in porous media. *Journal of Computational Physics* **227** (1): 706–727. doi: 10.1016/j.jcp.2007.08.012.

LeVeque, R.J. 1992. *Numerical Methods for Conservation Laws*. Basel, Switzerland: Lectures in Mathematics, Birkhaüser Verlag.

Li, X.S. 2005. An Overview of SuperLu: Algorithms, Implementation, and User Interface. *ACM Transactions on Mathematical Software (TOMS)* **31** (3): 302–325. doi: 10.1145/1089014.1089017.

Müller, S. and Stribia, Y. 2007. Fully Adaptive Multiscale Schemes for Conservation Laws Employing Locally Varying Time Stepping. *Journal of Scientific Computing* **30** (3): 493–531. doi: 10.1007/s10915-006-9102-z.

Naccache, P.F. 1997. A Fully-Implicit Thermal Reservoir Simulator. Paper SPE 37985 presented at the SPE Reservoir Simulation Symposium, Dallas, 8–11 June. doi: 10.2118/37985-MS.

Ortega, J.M. and Rheinboldt, W.C. 1970. *Iterative Solution of Nonlinear Equations in Several Variables*. New York: Academic Press.

Plewa, T., Linde, T., and Gregory, W.V. ed. 2003. *Adaptive Mesh Refinement—Theory and Applications. Proceedings of the Chicago Workshop on Adaptive Mesh Refinement Methods, Sept. 3–5, 2003*, Vol. 41. Berlin: Lecture Notes in Computational Science and Engineering, Springer.

Shroff, G.M. and Keller, H.B. 1993. Stabilization of Unstable Procedures: The Recursive Projection Method. *SIAM J. Numer. Anal.* **30** (4): 1099–1120. doi: 10.1137/0730057.

Watson, L.T. 1986. Numerical Linear Algebra Aspects of Globally Convergent Homotopy Methods. *SIAM Review* **28** (4): 529–545. doi: 10.1137/1028157.

Watson, L.T., Billups, S.C., and Morgan, A.P. 1987. Algorithm 652: HOMPACK: a suite of codes for globally convergent homotopy algorithms. *ACM Transactions on Mathematical Software* **13** (3): 281–310. doi: 10.1145/29380.214343.

Younis, R. 2009. Advances in Modern Computational Methods for Nonlinear Problems: A Generic Efficient Automatic Differentiation Framework, and Nonlinear Solvers that Converge All The Time. PhD thesis, Stanford University, Palo Alto, California.

Appendix A—Model Problems

We illustrate and apply the key ideas in this work using two reservoir simulation problems. The first is a 1D two-phase Buckley-Leverett model, and the second is a 2D two-phase-flow problem with compressibility and gravity.

Problem 1—Buckley-Leverett Model. We consider a two-phase Buckley-Leverett problem in 1D with gravity effects. Flow is through a domain with unit length, $x \in [0,1]$, and the unknown is the water saturation, $S(x, t)$. The governing equation in conservation form is written in Eq. A-1:

$$\begin{aligned} &S_t + f(S)_x = 0 \\ &S(x, t=0) = S_{\text{init}} \\ &S(x=0, t) = S_{\text{inj}}. \end{aligned} \quad \text{(A-1)}$$

In Eq. A-1, the initial saturation is denoted S_{init}, and the left boundary condition, S_{inj}, is fixed. The fractional flow function, $f(S)$, is defined in Eq. A-2, where the viscosity ratio, M^0, and the gravity number, Ng, are constant parameters:

$$f(S) = K_{rw}\left[\frac{1 - NgKr_o(S)}{Kr_w(S) + M^0 Kr_o(S)}\right]. \quad \text{(A-2)}$$

In this problem, we assume Hornarpour relations for relative permeabilty (Eqs. A-3 and A-4).

$$K_{rw} = K_{rw0}\left(\frac{S - S_{wr}}{1 - S_{wr} - S_{or}}\right)^{n_w}. \quad \text{(A-3)}$$

$$K_{ro} = K_{ro0}\left(\frac{1 - S - S_{or}}{1 - S_{wr} - S_{or}}\right)^{n_o}. \quad \text{(A-4)}$$

Note that, for an updip case, $Ng > 0$, or a down-dip case, $Ng < 0$, the fractional flow function may exhibit a local minimum or maximum, respectively. This leads to the possibility of countercurrent flow. In such cases, we refer to the extremum as the sonic point, a local maximum or minimum in the fractional flow, denoted f^* and occurring at a saturation S^*.

We apply a fully implicit discretization in time and a first-order upwind discretization in space. On a uniform mesh with N cells, the numerical saturation unknown at the nth timestep and the ith cell is denoted as S_i^n. The numerical scheme is written as in Eq. A-5.

$$S_i^{n+1} - S_i^n + \frac{\Delta t}{\Delta x}\left[F\left(S_i^{n+1}, S_{i+1}^{n+1}\right) - F\left(S_{i-1}^{n+1}, S_i^{n+1}\right)\right] = 0, i = 1, \ldots, N. \quad \text{(A-5)}$$

In Eq. A-5, the timestep size is denoted as Δt, the mesh spacing as $\Delta x = \frac{1}{N}$, and the numerical flux as F. We apply a Dirichlet boundary condition on the left of the domain and a second-order treatment of a free boundary condition on the right. These are numerically prescribed by Eqs. A-6 and A-7.

$$S_0^n = S_{\text{inj}}. \quad \text{(A-6)}$$

$$S_{N+1}^n = 2S_N^n - S_{N-1}^n. \quad \text{(A-7)}$$

For general fractional flow functions, it is necessary to apply an entropy-satisfying upwind condition for the numerical flux. This condition corresponds to the analytical solution of cell-face Riemann problems. The condition applied is described by Eq. A-8.

$$F(a,b) = \begin{cases} \min\limits_{a \le s \le b} f(s) & a \le b \\ \max\limits_{b \le s \le a} f(s) & \text{otherwise} \end{cases}. \quad \text{(A-8)}$$

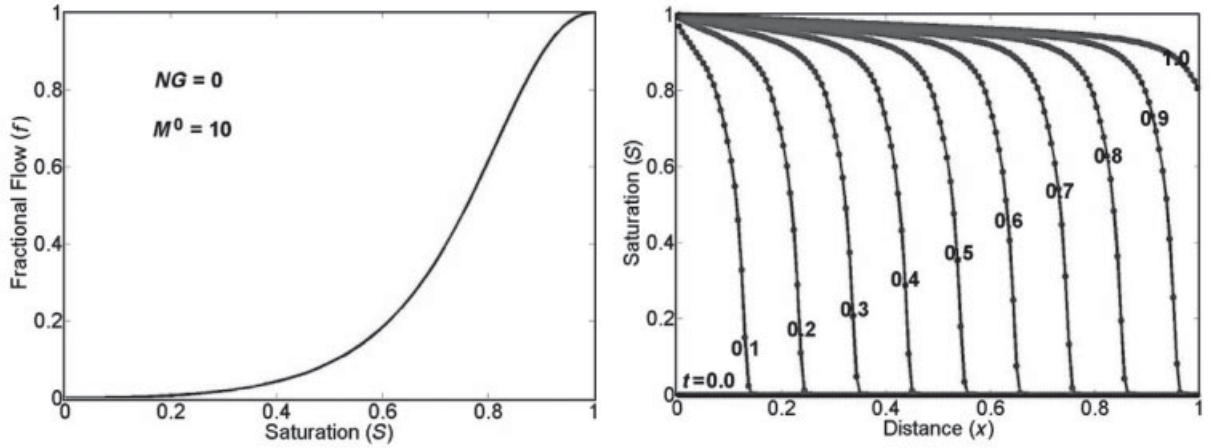
Note that, for fractional flows with sonic points, the numerical flux at a cell interface may be independent of both the left and right cell saturations when it is evaluated at the sonic point.

We illustrate fully implicit solutions for two cases of Problem 1. The first case is a horizontal piston-like displacement, and the second includes gravity effects and exhibits countercurrent flow. In both cases, the injection saturation is unity, $S_{\text{inj}} = 1$, and the relative permeability functions are quadratic with endpoints of zero and one.

In the horizontal case, the endpoint mobility ratio M^0 is chosen as 10, the gravity number Ng is chosen as 0, the initial saturation, S_{init}, is zero, and $N = 150$. **Figs. A-1a and A-1b** show the fractional flow curve and solution profiles for various timesteps, respectively.

In the down-dip case, the endpoint mobility ratio, M^0, is chosen as 0.5, and the gravity number, Ng, is chosen as −5 for a down-dip problem. The number of gridblocks used is $N = 150$. The initial condition has a saturation of 1 up to a distance of 0.34, and a saturation of 0 elsewhere. **Figs. A-2a and A-2b** show the fractional flow curve and solution profiles for various timesteps, respectively.

Problem 2—Compressible Flow With Gravity. The second problem involves two-phase compressible flow in 2D. Gravity effects are superposed with nonlinear relative permeability models and injection and production wells to stress the nonlinear solution aspect. The unknowns are pressure, $p(x, y, t)$, and water saturation,



(a) Fractional flow curve

(b) Solution profiles

Fig. A-1—Fractional flow curve (a) and saturation solution profiles (b) for a 1D Buckley-Leverett problem without gravity.

$S_w(x, y, t)$. The governing equations for the conservation of oil and water appear in Eqs. A-9 and A-10, respectively.

$$\left[\phi\left(1-S_w\right)\right]_t-\nabla\left[K\frac{K_{ro}}{\mu_o}\nabla\left(p+\gamma_o h\right)\right]=q_o. \quad \ldots\ldots\ldots\ldots (A\text{-}9)$$

$$\left[\phi S_w\right]_t-\nabla\left[K\frac{K_{rw}}{\mu_w}\nabla\left(p+\gamma_w h\right)\right]=q_w. \quad \ldots\ldots\ldots\ldots\ldots (A\text{-}10)$$

The compressibility in the problem is caused by a pressure-dependent porosity relation, as specified by Eq. A-11.

$$\phi=\phi_{\text{ref}}\left[1+c_r\left(p-p_{\text{ref}}\right)\right]. \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots (A\text{-}11)$$

Other parameters in this problem are

- A diagonal permeability tensor, K, which may be heterogeneous
- μ_w and μ_o, denoting constant water and oil viscosity, respectively
- K_{rw} and K_{ro}, denoting the water and oil relative permeability described by Eqs. A-3 and A-4, respectively
- γ_w and γ_o, denoting the constant water and oil gravimetric density
- h, denoting depth along the direction of gravity

We apply a fully implicit discretization with standard single-point phase-based upstream weighting. The oil-conservation equation is aligned with pressure, and the water equation with water saturation. We assume no-flow boundary conditions across the rectangular domain of dimensions L_x and L_y. A rate-controlled injector and bottomhole-pressure producer are introduced and are completed in single blocks. The initial condition may be transient, allowing the specification of an arbitrary initial saturation distribution. In such cases, the pressure distribution is initialized according to gravity and a specified pressure at the top of the reservoir.

Appendix B—Algorithmic Details of the CN Method

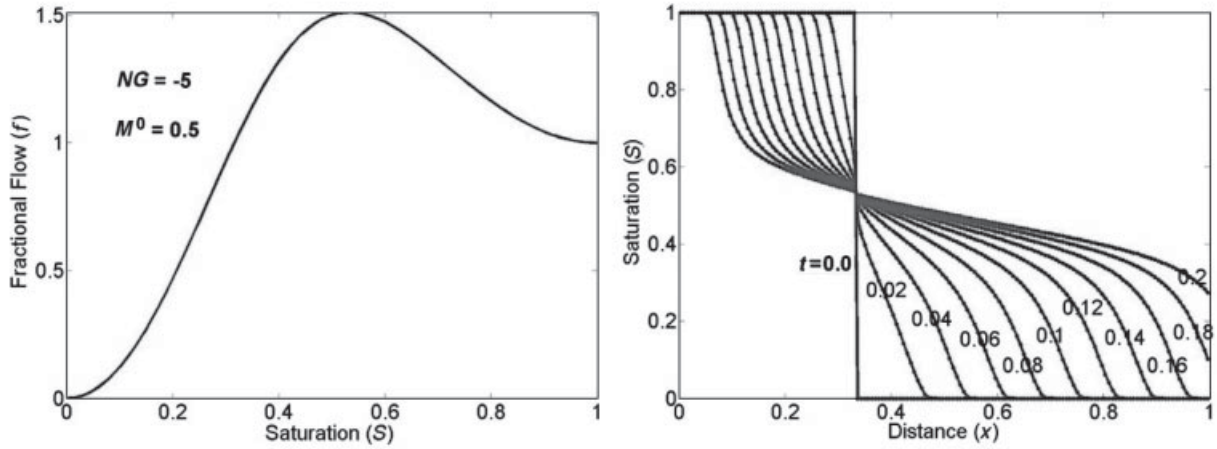
Algorithm 1 prescribes the general CN process to solve a target timestep Δt_{target} given an old state U^n, which may involve many unknowns.

Algorithm 1: CONTINUATION-NEWTON-STEP (U^n, Δt_{target}).

Require: $\Delta t_{\text{target}} \geq 0$ and $U^n \in \mathbb{R}^N$

Ensure: $F(U, \Delta t, U^n) = 0$, $0 < \Delta t \leq \Delta t_{\text{target}}$, and, niter $< N_{\max}$

niter $\leftarrow 0$
$\Delta t \leftarrow 0$



(a) Fractional flow curve

(b) Solution profiles

Fig. A-2—Fractional flow curve (a) and saturation solution profiles (b) for a down-dip 1D Buckley-Leverett problem.

$U \leftarrow U^n$
while niter < N_{max}, and $\Delta t < \Delta t_{target}$
  $\hat{\delta} \leftarrow$ COMPUTE-TANGENT(U, Δt, U^n)
  $\alpha \leftarrow$ SELECT-TANGENT-STEP-LENGTH $(U, \Delta t, \hat{\delta}, U^n)$
  $W \leftarrow U$
  for $i = 0$ to $i = N$ do
    $W(i) \leftarrow$ APPLEYARD-SAFE-UPDATE $\left[W(i), \alpha\hat{\delta}(i)\right]$
  end for
  $\Delta t_2 \leftarrow \alpha\hat{\delta}(N+1)$
  niter ← niter+1
  if IS-WITHIN-NEIGHBORHOOD(W, Δt_2, U^n)
    $U \leftarrow W$
    $\Delta t \leftarrow \Delta t_2$
  else
  /* *No tangent step length is admissible in convergence neighborhood* */
    U, $N_{newton} \leftarrow$ MODIFIED-APPLEYARD-NEWTON-CORRECTOR(U, Δt, U^n)
    niter ← niter+N_{newton}
  end if
end while
U, $N_{newton} \leftarrow$ MODIFIED-APPLEYARD-NEWTON-SOLVER(U, Δt, U^n)
niter ← niter +N_{newton}
return U, Δt, and, niter

Algorithm 1 uses several subalgorithms, some of which have already been discussed; the Appleyard safe update performs a cell-wise saturation update along the tangent direction, and the modified Appleyard-Newton corrector and solver performs a Newton process using the modified Appleyard heuristic. In Algorithm 1, local Newton corrections to bring points closer to the solution path are performed by a Newton solver except that a different, looser convergence tolerance is used. Specifically, the corrector convergence tolerance only needs to be smaller than or equal to the convergence neighborhood tolerance. The remaining subalgorithms are SOLVE-TANGENT, which computes the normalized tangent update vector, $\hat{\delta}$; SELECT-TANGENT-STEP-LENGTH, which performs a search along the tangent for the largest step that remains within the convergence neighborhood; and IS-WITHIN-NEIGHBORHOOD, which tests whether a point is inside the convergence neighborhood. These three subalgorithms are developed next.

Tangent Computation Algorithm. Algorithm 2 prescribes the details of computing the tangent vector at a point (U, Δt) close to the solution path emanating from the point (U^n, 0).

Algorithm 2: COMPUTE-TANGENT (U, Δt, U^n).

Require: $\Delta t \geq 0$ is a timestep size for state $U \in \mathbb{R}^N$ from the initial state $U^n \in \mathbb{R}^N$.

Ensure: $\hat{\delta} \in \mathbb{R}^{N+1}$ is the unit tangent to the augmented solution curve emanating from (U^n, 0) at the point (U^n, Δt).

Fix a positive constant C. The choice $C = \sqrt{\varepsilon_{mach}}$, where ε_{mach} is the machine precision, can be used for numerical conditioning.

$$C \leftarrow \max\left[C, \frac{1}{\left\|\frac{\partial}{\partial \Delta t} R\left(U; \Delta t, U^n\right)\right\|}\right]$$

$$\delta_u \leftarrow -CJ^{-1}\frac{\partial}{\partial \Delta t} R\left(U; \Delta t, U^n\right)$$

$$\delta_{\Delta t} \leftarrow C$$

$$\delta \leftarrow \left(\delta_u, \delta_{\Delta t}\right)^T$$

$$\hat{\delta} \leftarrow \frac{1}{\|\delta\|}\delta$$

return $\hat{\delta}$

Note that the value of the constant C is chosen as the inverse of the norm of the timestep derivative of the residual system. This improves the numerical scaling. The Jacobian matrix, J, need not be inverted explicitly, and any choice of efficient reservoir simulation linear solver can be used. Here, we focus on the nonlinear convergence aspects, and we apply a direct sparse solver with partial pivoting [see Li (2005)].

Prescribing a Convergence Neighborhood. Algorithm 3 describes the details of testing whether a given point (U, Δt) is within the convergence neighborhood about the solution path emanating from (U_0, 0). The choice of residual tolerance cut-off is subjective. Accepting the locally quadratic convergence rate of Newton's method, we use a tolerance that is one or two orders of magnitude looser than that required to judge if an iterate is a valid solution.

Algorithm 3: IS-WITHIN-NEIGHBORHOOD(U, Δt, U_0).

Require: $\Delta t \geq 0$ is a timestep size for state $U \in \mathbb{R}^N$ from the initial state $U_0 \in \mathbb{R}^N$.

Ensure: Returns whether U is within the convergence neighborhood at timestep Δt from state U_0.

Fix a positive tolerance ε_{cntrol}. A typical choice may be $\varepsilon_{cnrtol} = \sqrt{\varepsilon_{rtol}}$, where ε_{rtol} is a residual tolerance used for a Newton correction process.
  if $\left\|R\left(U, \Delta t; U_0\right)\right\| < \varepsilon_{cnrtol}$ **then**
    return true
  else
    return false
  end if

Step Length Selection Algorithm. The algorithm used to solve the univariate problem is derivative-free. Each iteration requires simply the evaluation of the residual. We apply a backtracking approach that starts by evaluating whether α_{min} results in an update that lies within the convergence neighborhood. If this is not the case, the search is terminated, and Newton corrections are triggered. Otherwise, backtracking from α_{max}, we search for the first step length that produces an update within the neighborhood. The maximum number of backtracking steps is a parameter of the algorithm. Increasing this quantity may result in a solution with fewer tangent steps at the expense of more residual evaluations performed per tangent step. Algorithm 4 describes the details of this process.

SELECT-TANGENT-STEP-LENGTH $\left(U, \Delta t, \hat{\delta}, U^n\right)$.

Fix a maximum number of backtracking iterations; $\text{Iter}_{max} \geq 1$.
Fix a minimal timestep size advancement per tangent step, $\Delta t_{min} > 0$. This can be based on a suitable CFL number such as 1.
Fix a maximal timestep size advancement per tangent step, $\Delta t_{max} > \Delta t_{min}$. A typical CFL number may be 10.

$$\alpha_{min} \leftarrow \frac{\Delta t_{min}}{\hat{\delta}(N+1)}$$

$$\alpha_{max} \leftarrow \frac{\Delta t_{max}}{\hat{\delta}(N+1)}$$

$$\Delta \leftarrow \frac{\alpha_{max} - \alpha_{min}}{\text{Iter}_{max}}$$

$\alpha \leftarrow \alpha_{max}$
do
  for $i = 0$ to N *do*
    $U^*(i) \leftarrow$ APPLEYARD-SAFE-UPDATE$\left[U(i), \alpha\hat{\delta}(i)\right]$
  end for
  $\Delta t^* \leftarrow \Delta t + \alpha\hat{\delta}(N+1)$
  is-converged ← IS-WITHIN-NEIGHBORHOOD (U^*, Δt^*, U^n)
  $\alpha \leftarrow \alpha - \Delta$
until $\alpha < \alpha_{min} - \Delta$ or is-converged = **true**
return α

Appendix C—Theoretical Development of Localization for Hyperbolic Transport in 1D

For the purpose of motivating our approach, we restrict our attention in this section to quasilinear scalar conservation laws in 1D (Eq. C-1). The flux function f is assumed to be differentiable with derivative f' and may be spatially dependent and generally nonlinear; it may also have sonic points and is not necessarily convex.

$$u(x,t)_t + f(x,u)_x = 0 . \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (C\text{-}1)$$

We are interested in square nonlinear systems arising from implicit two-level approximations of the Godunov type [see, for example, LeVeque (1992)]. A general form for the residual is written as

$$R^{(\nu)} \equiv \left[I + c_x \left(F^R - F^L \right) \right] U^{(\nu)} - U^n = 0, \quad \ldots\ldots\ldots\ldots\ldots (C\text{-}2)$$

where the mesh ratio $c_x = \Delta t/\Delta x \geq 0$ is fixed and ν is the nonlinear iteration index. Note that, in the definition of the residual, we restrict the analysis to first-order upwind approximations, which can generally be written as

$$F_i^R = \begin{cases} \min\limits_{U_i \leq U \leq U_{i+1}} f_{i+1/2}(U) & U_i \leq U_{i+1} \\ \max\limits_{U_{i+1} \leq U \leq U_i} f_{i+1/2}(U) & \text{otherwise} \end{cases}, i = 1,2,\ldots N. \quad \ldots\ldots\ldots (C\text{-}3)$$

Note that $F_i^L = F_{i-1}^R$. For a given nonlinear iteration, ν, we have at hand an iterate, $U^{(\nu)}$, and we compute a corresponding residual, $R^{(\nu)}$, and Jacobian, $J^{(\nu)}$. From now on, iteration index superscripts will be dropped, and all terms are assumed to be at the νth nonlinear iteration of a timestep. In this notation, the corresponding Newton step is $\delta \equiv -J^{-1}R$.

By linearity, we can also write the Newton step as a combination of substeps, each resulting from the action of the inverse Jacobian on a single component of the residual. That is, let $R = R_1 + \ldots + R_N$ be such that

$$R_j^{(i)} = \begin{cases} R^{(i)} & i = j \\ 0 & \text{otherwise} \end{cases}, \; 0 < i \leq N,$$

where the superscript index i is the block index and the subscript index j is that of the cell with a nonzero residual entry. Subsequently, the Newton step can be written as $\delta = \delta_1 + \ldots + \delta_N$, where $\delta_j \equiv -J^{-1}R$. Each Newton substep represents the state update over an iteration caused by a nonzero residual entry in a single cell. We will show next that each of these substeps essentially propagates the isolated nonzero entry in the residual according to the local upwind direction and the local wave speeds. More precisely, we derive a quantitative relation for the numerical range of influence caused by a given nonzero residual entry. This is described first for problems without sonic points. We then generalize the scheme to countercurrent-flow problems involving sonic points. Finally, we combine this information to formulate a quantitative relation for the total range of influence.

Flux Functions Without Sonic Points. Given an iterate, we first suppose that the flux function has no extremal points over the cell-state range. In this case, the upwind directions will be unidirectional throughout the entire domain. If the flux function has nonnegative derivatives, $f' \geq 0$, the resulting flow is from left to right (upwind directions are to the left). Alternatively, for $f' \leq 0$, the flow is from right to left. We derive in detail the main results assuming $f' \geq 0$ and state the corresponding result in the other case.

Denoting the local linearized CFL number as $\sigma_i \equiv c_x \left| f_i' \right|$, the ith entry in the resulting Newton step is given by

$$\delta^{(i)} = -\frac{1}{1+\sigma_i}\left[R^{(i)} - \sigma_{i-1}\delta^{(i-1)} \right], 0 < i \leq N .$$

The magnitude of the Newton step in a given cell depends on, at most, the residual values in the cells upstream of it. We can exploit this fact to quantitatively identify the numerical range of influence caused by a nonzero residual entry in each cell independently.

For a given cell $0 < j \leq N$, we can write the corresponding Newton substep $\delta_j = -J^{-1}R_j$ as

$$\delta_j^{(i)} = \begin{cases} 0 & 0 \leq i < j \\ -\zeta_j \boldsymbol{R}^{(j)} & i = j \\ -\left(\prod_{k=j}^{i-1} \theta_k \right) \zeta_i \boldsymbol{R}^{(j)} & j < i < N \end{cases}, \; 0 < i \leq N, \quad \ldots\ldots (C\text{-}4)$$

where we have the local scaling constant, ζ_i, written as

$$\zeta_i \equiv \frac{1}{1+\sigma_i} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (C\text{-}5)$$

and the local attenuation ratios, θ_k, given by

$$\theta_k \equiv \frac{\sigma_k}{1+\sigma_k}. \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (C\text{-}6)$$

The Newton substep does not modify the iterate state in cells upstream of the disturbed cell j. The cells downstream of j will be updated by a quantity that is a scaled version of the update at the disturbance source. The question we answer here is how far out downstream into the domain are the effects of the disturbance felt.

Because $\sigma_i \geq 0$ for $0 < i \leq N$, both the local attenuation ratios and the scaling constants are bounded by 1. Denoting the maximum and minimum CFL numbers over all cells downstream of j as $\sigma_{\max,j}$ and $\sigma_{\min,j}$, we derive bounds on the local scaling constants as

$$0 < \zeta_i \leq \zeta_{\max,j} \equiv \frac{1}{1+\sigma_{\min,j}} \leq 1 \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots (C\text{-}7)$$

and on the local attenuation ratios as

$$0 \leq \theta_i \leq \theta_{\max,j} \equiv \frac{\sigma_{\max,j}}{1+\sigma_{\max,j}} < 1 . \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots (C\text{-}8)$$

We are interested in identifying the propagation rate through the Newton substep. Using the bounds in Eqs. C-7 and C-8, a conservative upper bound on the magnitude of the substep update in any cell downstream of the disturbance is given as

$$\left| \delta_j^{(i)} \right| \leq \left(\theta_{\max,j} \right)^{i-j} \zeta_{\max,j} \left| R^{(j)} \right|, j \leq i \leq N. \quad \ldots\ldots\ldots\ldots\ldots\ldots (C\text{-}9)$$

The alternative case, $f_i' \leq 0$, is treated similarly. For a given Newton substep caused by a disturbance in cell j, we have that the magnitude of the update will be bounded as

$$\left| \delta_j^{(i)} \right| \leq \left(\theta_{\max,j} \right)^{j-i} \zeta_{\max,j} \left| R^{(j)} \right|, 1 \leq i \leq j . \quad \ldots\ldots\ldots\ldots\ldots\ldots (C\text{-}10)$$

Sonic Points and Countercurrent Flows. More generally, the upwind direction may not be uniform throughout the domain. In particular, the flux function may map from extremal points over the range of state values in a given iterate. With single-point upwind schemes, this implies that at any cell face, the upwind flux of a particular phase may be evaluated at either left-, right-, or sonic-point states. In the first two cases, the upwinding implies a coupling from the upstream cell to one or more cells downstream. In the case of a sonic point, numerically, no coupling occurs over the timestep (i.e., the zero speed characteristic of a Riemann fan is independent of the left and right states). Subsequently, it is easy to show that, for general states and a given substep of interest, one of Eqs. C-9 and C-10 would hold.

Fig. C-1 depicts a mesh over which a Newton iterate has a varying upwind direction throughout the domain. Within the disjoint closed subdomains labeled Ω_1, and Ω_4, the flow direction is positive.

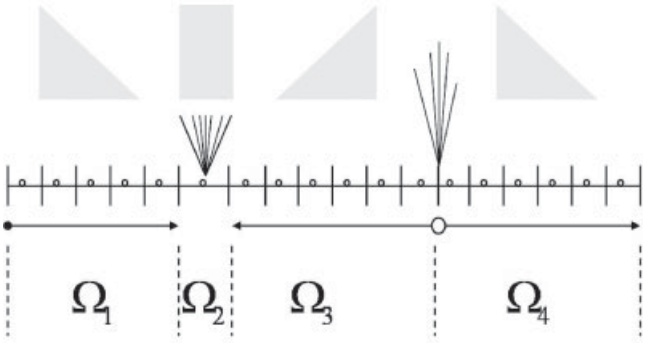


Fig. C-1—Sketch of a 1D mesh over which an iterate involves changing upwind directions.

Within Ω_3, it is negative. Finally, the state within subdomain Ω_2 is that of a Riemann fan over the timestep interval. Subsequently, a nonzero residual entry in some cell $j \in \Omega_1$ or $j \in \Omega_4$, can give rise to nonzero correction entries in the Newton substep, $\delta_j^{(i)}$, only within cells $\{i : i \in \Omega_{1,4}, i \geq j\}$. Within these cells, the precise magnitude of the corrections satisfy Eq. C-4. Moreover, the substep correction entries are bounded as in Eq. C-9, where, now, the maximum and minimum local CFL numbers are to be taken over the set of cells $\{i : i \in \Omega_{1,4}, i \geq j\}$. Similarly, for a nonzero residual entry in $j \in \Omega_3$, the effects can be felt only through $\{i : i \in \Omega_3, i \leq j\}$ and the bounds in Eq. C-10 hold. Finally, the effects of a nonzero residual in Ω_2 are isolated to that single cell.

Algebraic Localization Algorithm. Having a priori conservative bounds that are easily computable, we can specify a desired absolute tolerance $\varepsilon_{\text{tol}} > 0$, for which we wish to determine how far downstream of the disturbance will the magnitude of the Newton substep components exceed the tolerance. That is, we wish to determine an index $j \leq i_{\text{range}} \leq N$, such that for each $k \in \{j, \ldots, i_{\text{range}}\}$, we have $|\delta_j^{(l)}| \leq \varepsilon_{\text{tol}}$. Indeed, such a conservative bound is easily derived as

$$\hat{i}_{\text{range}} \equiv \left\lceil \frac{\ln(\varepsilon_{\text{tol}}) - \ln\left(\zeta_{\max,j} |R^{(j)}|\right)}{\ln(\theta_{\max,j})} \right\rceil + j,$$

$$i_{\text{range},j} = \max\left[j, \min\left(\hat{i}_{\text{range}}, N\right)\right].$$

Algorithmically, the absolute error tolerance, $\varepsilon_{\text{tol}} > 0$, can be set using the same criteria used in controlling inexact linear solvers. That is, we may apply all of the available theoretical machinery for relating the overall nonlinear convergence rate with the accuracy of computing Newton steps. What is unique about our range-of-influence approach is that, even for very tight tolerances $\varepsilon_{\text{tol}} \to 0$, the savings in computational effort can be substantial.

Finally, this analysis provides a strategy to localize computations for each nonlinear iteration. Given the iterate, we can simply determine the index sets $\mathcal{I}_{\text{active},j} = \{i : j \leq i \leq i_{\text{range},j}\}$ independently for each $j \in \{1, \ldots, N\}$. Cells indexed by a set will contain nonnegligible Newton substeps. Cells indexed by the complement sets will have negligible Newton changes. The union of all active index subsets $\mathcal{I}_{\text{active}} = \bigcup_{j=1}^{N} \mathcal{I}_{\text{active},j}$ flags the cells that must be updated. The reduction procedure is a standard column and row extraction from the Jacobian matrix and corresponding residual elements followed by a linear solve.

R.M. Younis is a PhD candidate in the Department of Energy Resources Engineering at Stanford University. He is interested in advancing the state of the art in nonlinear algorithms and software infrastructure to enable reliable and predictive large-scale simulation of emerging subsurface applications. Younis holds a BE degree from McGill University, and an MS degree in petroleum engineering and an MS degree in scientific computing and computational mathematics from Stanford University. He was awarded the Ramey Fellowship for achievement in research. **Hamdi Tchelepi** is associate professor of energy resources engineering at Stanford University. Previously, he was with Chevron Energy Technology Company for 10 years. Tchelepi holds a PhD degree in petroleum engineering from Stanford. He codirects the Reservoir Simulation Industrial Affiliates program at Stanford and is interested in developing multiscale formulations, scalable linear and nonlinear solvers with application to reservoir simulation and subsurface CO_2 sequestration. **Khalid Aziz** is the Otto N. Miller Professor of Earth Sciences and a professor of petroleum engineering at Stanford University. Before coming to Stanford in 1982, he was a professor of chemical and petroleum engineering at the University of Calgary. Aziz holds a BSE degree in mechanical engineering from the University of Michigan, BS and MS degrees in petroleum engineering from the University of Alberta, and a PhD degree in chemical engineering from Rice University. He has served SPE in many capacities, including as director and distinguished lecturer. He is an honorary member of SPE and a member of the National Academy of Engineering.