A Sequential Semi-Implicit Algorithm for Computing Discontinuous Flows in Porous Media

M. A. Vegas-Landeau[SPE], R. Propp, T.W.Patzek[SPE] and P. Colella
University of California at Berkeley

A novel numerical algorithm for computing incompressible, discontinuous, two-phase flows in two-dimensional, inhomogeneous, and isotropic porous media is presented. The algorithm uses Colella *et al.'s* hybrid sequential semi-implicit approach[6] for both accuracy and efficiency of the calculations. The explicit part uses a high-order Godunov[11] scheme with a modified Van Leer geometrical slope limiter, similar to those used in shock dynamics. The implicit part is a two-step solver: the first step is a Crank-Nicolson saturation solver and the second one is a Poisson solver for the phase pressure. Both use fast multilevel multigrid solvers with the number of operations of the order of $\mathcal{O}[N\log(N)]$, where N is the number of grid points. Two numerically stiff reservoir-engineering problems are presented to demonstrate the low numerical dispersion and second-order accuracy of our method.

Introduction

In this work we present a semi-implicit numerical method for solving the equations of incompressible flow in a porous medium.

The starting point for our method is a IMPES-type splitting based on a total velocity, in which the pressure satisfies an elliptic PDE derived from Darcy's law, and the saturation satisfies a nonlinear advection-diffusion equation. Our approach is a generalization of the method of Bell, Colella, and Trangenstein extended to the case of nonzero capillary pressure. We use a predictor-corrector approach based on an explicit differencing of the advective terms and a Crank-Nicholson type differencing of the parabolic terms, following the ideas of Bell, Colella, and Glaz[6]. The resulting method is fully second-order accurate in space and time, and has the property that the only linear systems that must be solved are those arising from well-behaved discretizations of second-order positive-semi-definite elliptic equations.

Paper first presented at the SPE Annual Technical Conference & Exhibition (Student Paper Contest) held in Denver,CO, U.S.A., 1996.

Received for review, February 7, 1997
Accepted for publication, July 8, 1997
Camera-ready copy received, December 14, 1997

For that reason, we are able to use efficient iterative methods based on multigrid iteration to solve those linear systems.

Although the restriction of incompressibility is a significant simplification of the conditions in a real petroleum reservoir, it gives some indication of the behavior of such a method on flows with discontinuities.

Mathematical Model

Theory. In this section, we will describe two-phase, two-dimensional, transient flow of an incompressible liquid (l) and gas (g). The saturation equation is written as a function of the total velocity, and not of the phase velocities. Although we have treated the gas as an incompressible phase, the BCT schemes were designed for compressible flows, a fact that will be used in future extensions of the algorithm.

The superficial velocity of each phase in porous media can be expressed using Darcy's law

$$\vec{u}_p = -\lambda_p\left(\vec{\nabla}P_p + \gamma_p\vec{\nabla}y\right) \quad \text{.....(1)}$$

where $\vec{u}_p$ is the phase velocity; λ_p is the phase mobility; $\gamma_p = \rho_p g$ is the specific gravity term; and $\vec{y}$ is a unit vector parallel to $-\vec{g}$. λ_p is only a function of the phase saturation s_p. The porous medium is described by an isotropic permeability $k(x,y)$ and porosity $\phi(x,y)$, both arbitrary functions of position.

Mass conservation requires that:

$$\frac{\partial(\phi\rho_p s_p)}{\partial t} + \vec{\nabla}\cdot(\rho_p u_p) = 0 \quad \text{.....(2)}$$

subject to the usual constraint:

$$s_l + s_g = 1 \quad \text{.....(3)}$$

The capillary pressure is defined:

$$P_c = P_g - P_l \quad \text{.....(4)}$$

hence:

$$\vec{\nabla}P_c = \vec{\nabla}P_g - \vec{\nabla}P_l \quad \text{.....(5)}$$

The total velocity is the sum of velocities of each phase at a given location:

$$\vec{u}_T = \vec{u}_l + \vec{u}_g \quad \text{.....(6)}$$

Since both the phases and the matrix are incompressible, $\vec{\nabla}\cdot(\vec{u}_T) = 0$, i.e., the total velocity is divergence free. From this we can build an initial-condition pressure equation. Substituting the Darcy velocity in the total velocity equation (6), yields:

$$\vec{\nabla}\cdot\left[\lambda_g\left(\vec{\nabla}P_g + \gamma_g\vec{\nabla}y\right)\right] + \vec{\nabla}\cdot\left[\lambda_l\left(\vec{\nabla}P_l + \gamma_l\vec{\nabla}y\right)\right] = 0 \quad \text{...(7)}$$

With the help of Eqs. (1-6), we derive the governing saturation equation. First, we substitute the Darcy velocities (1) into the definition of the total velocity (6):

$$\vec{u}_l = \frac{\lambda_l}{\lambda_l+\lambda_g}\vec{u}_T + \frac{\lambda_l\lambda_g}{\lambda_l+\lambda_g}\left[\vec{\nabla}P_c - (\gamma_g - \gamma_l)\vec{\nabla}y\right] \quad \text{.....(8)}$$

Second, we substitute $\vec{u}_l$, Eq. (8), into the mass conservation equation (2):

$$\phi\frac{\partial s_l}{\partial t} + \vec{\nabla}\cdot\left[\left(\frac{\lambda_l\lambda_g}{\lambda_l+\lambda_g}\right)(\gamma_g - \gamma_l)\vec{\nabla}y + \frac{\lambda_l^2}{\lambda_l+\lambda_g}\vec{u}_T\right]$$

$$= -\vec{\nabla}\cdot\left(\frac{\lambda_l\lambda_g}{\lambda_l+\lambda_g}\vec{\nabla}P_c\right) \quad \text{.....(9)}$$

Eq. (9) is arranged into a flux function form

$$\phi\frac{\partial s_l}{\partial t} + \vec{\nabla}\cdot\vec{F} = -\vec{\nabla}\cdot\left[W\vec{\nabla}P_c\right] \quad \text{.....(10)}$$

where

$$\vec{F} = \frac{\lambda_l\left[\vec{u}_T - (\gamma_l - \gamma_g)\vec{\nabla}y\right]}{\lambda_l+\lambda_g} \quad \text{.....(11)}$$

is the flux vector, and

$$W = \frac{\lambda_l\lambda_g}{\lambda_l+\lambda_g} \quad \text{.....(12)}$$

The flux vector is central to our formulation, because it contains useful information that will be used to find many flow properties.

To complete our sequential mathematical model, we need an expression for the gas pressure:

$$\vec{\nabla}\cdot\left[(\lambda_g+\lambda_l)\vec{\nabla}P_g\right] = \vec{\nabla}\cdot(\lambda_g\gamma_g + \lambda_l\gamma_l) + \vec{\nabla}\cdot(\lambda_l\vec{\nabla}P_c)$$
$$\text{.....(13)}$$

The phase coupling has been "eliminated" by the use of the total velocity definition. In this, way we can compute the hyperbolic and the elliptic components of our problem separately, and introduce the correct mathematical, geometrical and physical constructs to solve accurately our discontinuous flow problem. The elliptic and hyperbolic equations are different in nature and require different numerical schemes[12].

The capillary pressure is described by the logarithmic Leverett J-function, Grosser[10,13]:

$$P_c(s) = \left(0.48 + 0.036\log\left(\frac{1-s_l}{s_l - s_{l,irr}}\right)\right)\sigma_{\lg}\cos(\Theta)\sqrt{\frac{\phi}{k}}$$
$$\text{.....(14)}$$

where $\sigma_{\lg}$ is the liquid-gas interfacial tension; Θ is the liquid contact angle.

The relative permeabilities are described by Corey's power-law model:

$$k_{rl} = k_{rl}^O\left(s_l - s_{l,irr}\right)^{e_l} \quad \text{.....(15)}$$

$$k_{rg} = k_{rg}^O\left(1 - s_l - s_{g,irr}\right)^{e_g} \quad \text{.....(16)}$$

with $e_l = 2$, $e_g = 1.5$, $s_{p,irr}$ is the irreducible saturation of phase p.

Numerical Algorithm

Our algorithm is shown schematically in **Figure 1**.

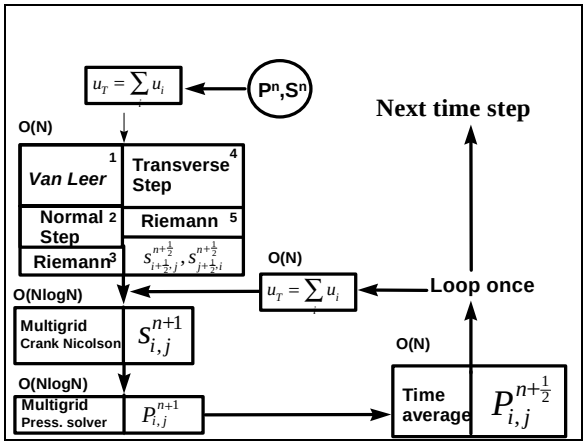

Figure 1: **Outline of the High Order Sequential Semi-implicit Algorithm.**

We discretize on a rectangular grid in each direction. Our algorithm uses both cell-centered and edge-centered variables. Each variable is placed at its physical position, i.e., saturations, pressures and capillary pressures are cell-centered, and velocities are edge-centered. However, there are still several problems. In order to evaluate the phase velocities, we must compute the phase mobilities that are functions of saturations (cell-centered).

We must choose the correct saturation for evaluating the mobilities at the edges of the cell. This problem has been studied extensively in the literature[2]. Traditionally, it has been suggested that the appropriate value should come from an upwind scheme. Although this approach is correct, it is only first-order accurate. Darcy's law implies that mobilities have the same centering as velocity, which is edge-centered. Fos this we must choose the correct saturation for evaluating the mobilities at the edges of the cell.

Our goal is to develop a higher-order scheme for better accuracy. However, choosing a higher-order scheme to determine the saturation at the edges, can violate entropy conditions[9]. This violation does not occur when the flow is continuous, i.e., the saturations and their higher derivatives are continuous. It occurs however when the flow is discontinuous, because the governing differential equation no longer represents accurately the physical problem. In the latter situation, we must use weak formulations and entropy conditions that have been extensively studied in the shock dynamics literature[15]. Only low-order schemes do not violate the Enquist-Osher conditions[9, 18]. Hence the numerical algorithm developed here uses a high-order approximation where the flow is continuous and a low-order approximation at each discontinuity, i.e., at a moving front.

Now we can introduce the high-order Godunov scheme. First, we approximate $u^n_{i,j}$ as the average of $\vec{u}$ over a finite difference cell centered at $(i\Delta x, j\Delta y)$, such that:

$$u^n_{i,j} = \frac{1}{\Delta x \Delta y}\int_{(i-\frac{1}{2})\Delta x}^{(i+\frac{1}{2})\Delta x}\int_{(j-\frac{1}{2})\Delta y}^{(j+\frac{1}{2})\Delta y} u(x, y, t^n)\,dxdy \quad \text{.....(17)}$$

Second, time is discretized by a time step $\Delta t = t^{n+1} - t^n$. A high-order predictor-corrector method to advance the solution from time t^n to time t^{n+1} is used. The predictor step estimates $s^{n+\frac{1}{2}}$. The corrector uses this estimate to predict the saturation and the pressure at time t^{n+1}. We do this in several steps (cf. **Figure 1)** as follows:

1. Estimate the phase velocities.
   a) Compute Darcy's velocities using second-order averaging for the phase mobilities.
   b) Compute total velocities.

2. High-order predictor step (Godunov algorithm).
   a) Compute High Order Van Leer slopes.
   b) Compute the left and right states of $\tilde{s}^{n+\frac{1}{2}}$ with normal and transverse predictor steps.
   c) Estimate the correct $\tilde{s}^{n+\frac{1}{2}}$ with a Riemann problem solver.
   d) Advance in time from $t^{n+\frac{1}{2}}$ to t^{n+1}.
   e) Compute $\tilde{P}^{n+1}$ implicitly using the pressure equation.

3. Corrector step.
   a) Compute average pressure,
   $$P^{n+\frac{1}{2}} = \frac{P^n + \tilde{P}^{n+1}}{2} \quad \text{.....(18)}$$
   b) Recompute Darcy's Law and the total phase velocities.
   c) Compute S^{n+1} using the saturation advection-diffusion equation.
   d) Compute P^{n+1} using the pressure equation.

These steps will be explained in the following sections.

Darcy velocities. As mentioned in the previous section, velocities are calculated using Darcy's law. For the p^{th} phase this is discretized as:

In the $\vec{x}$-direction:

$$u_{p:i+\frac{1}{2},j} = -\lambda_{p:i+\frac{1}{2},j}\left(\frac{P_{p:i+1,j} - P_{p:i,j}}{\Delta x}\right) \quad \text{.....(19)}$$

In the $\vec{y}$ -direction:

$$u_{p:i,j+\frac{1}{2}} = -\lambda_{p:i,j+\frac{1}{2}}\left(\frac{P_{p:i,j+1} - P_{p:i,j}}{\Delta y} + \gamma_{p:i,j+\frac{1}{2}}\right) \quad \text{.....(20)}$$

where

$$\lambda_{p:i+\frac{1}{2},j} = \frac{1}{2}\left(\lambda_{p:i+1,j} + \lambda_{p:i,j}\right) \quad \text{.....(21)}$$

that is, we *do not* upwind the mobilities, but rather use a second-order averaging.

The total edge-centered velocities can be expressed as $\vec{u}_T = \vec{u}_l + \vec{u}_g$, cf. Eqs. (19-20).

Fourth-Order Van Leer Slopes. The Van Leer slope limiting[16,17,18] is geometric by nature. The basic idea is to generalize Godunov's method by replacing the piecewise constant representation of the solution by a more accurate representation, say, piecewise linear. Usually the oscillations that arise with most high-order schemes (Gibbs's phenomena) can be interpreted geometrically as being caused by a poor choice of slopes.

We can rectify this problem by applying a Van Leer slope limiter that reduces the value of the slope near the discontinuities or extreme points[15]. In this work, we will calculate the Van Leer slopes for the whole domain, assuming that the flux function is fully convex and then correct in those subdomains in which the flux is non-convex in order to achieve proper vectorization and/or parallelization of our algorithm.

For convex fluxes, we begin by calculating the Van Leer slopes:

$$\Delta_{\lim} = \begin{cases} 2\min\left(\left|s_{i+1,j} - s_{i,j}\right|, \left|s_{i,j} - s_{i-1,j}\right|\right) & if\ \left(s_{i+1,j} - s_{i,j}\right)\left(s_{i,j} - s_{i-1,j}\right) > 0 \\ 0 & otherwise \end{cases}$$

.....(22)

$$\Delta^{x,f} = \min\left(\frac{1}{2}\left|s_{i+1,j} - s_{i-1,j}\right|, \Delta_{\lim}\right) sign\left(s_{i+1,j} - s_{i-1,j}\right). \quad \text{(23)}$$

$$\Delta^{x}_{\lim} = \min\left(\frac{2}{3}\left|s_{i+1,j} - s_{i-1,j} - \frac{1}{4}\left(\Delta^{x,f} s_{i+1,j} + \Delta^{x,f} s_{i-1,j}\right)\right|, \Delta_{\lim}\right)$$

.....(24)

$$\Delta^{VL}_{\lim} = \Delta^{\lim}_{x} sign\left(s_{i+1,j} - s_{i-1,j}\right) \quad \text{.....(25)}$$

For those parts of our domain in which flow is not convex, the slopes must be corrected. In order to determine where the flow is convex, we need to know if we are in the neighborhood of a discontinuity. For this, we can estimate the following function:

$$\Psi^* = \frac{\left|s_{i+1,j} - s_{i-1,j}\right|}{\max\left(\left|s_{i+1,j} - s_{i-1,j}\right|, \left|s_{i+2,j} - s_{i-2,j}\right|, \varepsilon\right)} \quad \text{.....(26)}$$

The function Ψ^* works as follows. If s behaves linearly, Ψ^* is equal to 0.5, but in the presence of a [0,1] discontinuity, Ψ^*=1. This means that Ψ^*<0.5 is a smooth function. $\Psi^* \cong 1$ indicates the presence of a discontinuity. Let

$$\Psi = \frac{\Psi^* - \Psi^H}{\Psi^L - \Psi^H} \quad \text{.....(27)}$$

where Ψ^H = 0.75, Ψ^L = 0.5 These values are not exact and were obtained from numerical experimentation.

Define:

$$\theta^* = \begin{cases} 1 & if\ \Psi < \Psi^L \\ \Psi & otherwise \end{cases} \quad \text{.....(28)}$$

$$\theta = \begin{cases} 0 & if\ \Psi > \Psi^H \\ \theta^* & otherwise \end{cases} \quad \text{.....(29)}$$

θ is a factor by which we multiply our high-order slopes. In the presence of a front, θ=0, which means that the slope is zero. Soon we will see that this yields a low-order flux approximation in the scheme. On the other hand, θ=1 indicates that the flow is smooth enough and a high-order approximation can be used.

For $\Psi^L < \Psi < \Psi^H$, a linear interpolation is used.

Now that we have detected the presence of a moving front, we need to estimate the presence of convexity changes to complete this step. The convexity in the flux can be computed by calculating the change of sign in the second derivative of flux function. If there is a change, $\Delta^{VL}_{\lim} = \theta \Delta^{VL}_{\lim,convex}$, otherwise $\Delta^{VL}_{\lim} = \Delta^{VL}_{\lim,convex}$.

The Normal and Transverse Predictors. Expanding $s^{n+\frac{1}{2}}_{i+\frac{1}{2},j}$ into a Taylor series derives the normal predictor step:

$$\hat{s}^{n+\frac{1}{2}}_{i+\frac{1}{2},j} = s^{n}_{i,j} + \frac{\partial s}{\partial x}\frac{\Delta x}{2} + \frac{\partial s}{\partial t}\frac{\Delta t}{2} \quad \text{.....(30)}$$

The normal step is in the direction of flow and the transverse step is the one perpendicular to it. In the normal direction.. We have dropped the phase subscript, because only one of the phase saturations needs to be solved for. After substituting this expression in the advection-diffusion equation (10) and discretizing, we obtain:

$$\hat{s}^{n+\frac{1}{2}}_{i+\frac{1}{2},j} = s^n_{i,j} + \frac{\partial s}{\partial x}\frac{\Delta x}{2} - \frac{\Delta t}{2\phi}\left(\vec{\nabla}\cdot\left(W^n\vec{\nabla}P^n_c\right) + \left(\frac{\partial F_y}{\partial y}\right)^n\right) \quad (31)$$

where the approximation

$$\left.\frac{\partial F_y}{\partial y}\right|_{i+\frac{1}{2},j} \approx \frac{F_{y:i+\frac{1}{2},j+1} - F_{y:i+\frac{1}{2},j-1}}{\Delta y} \quad (32)$$

is used and

$$\frac{\partial s}{\partial x} \approx \frac{\Delta s^{VL}}{\Delta x} \quad (33)$$

Hence, we approximate the saturation slope with the high-order Van Leer slope. After combining Eqs. (31)-(33) and rearranging, we get:

$$\hat{s}^{n+\frac{1}{2}}_{i+\frac{1}{2},j} = s^n_{i,j} + \frac{1}{2}\left(1 - \frac{\Delta t}{\phi\Delta x}\max\left(0, \frac{\partial F_y}{\partial s}\right)\right)\Delta s^{VL} - \frac{\Delta t}{2\phi}\left(\vec{\nabla}\cdot\left(W^n\vec{\nabla}P^n_c\right) + \frac{F_{y:i+\frac{1}{2},j+1} - F_{y:i-\frac{1}{2},j+1}}{\Delta y}\right) \quad (34)$$

The term $\max(0, \partial F / \partial s)$ is included to guarantee that the waves are moving toward the cell edge at which the solution is expanded, and not in the opposite direction. It is interesting to observe that this scheme reduces to upwind when the limiter is equal to zero, which occurs in the presence of a moving front. In this way, we do not violate the Osher-Enquist entropy conditions, and we do guarantee a physical solution. For the transverse step, we use a similar procedure taking into account that the normal component of the flux is no longer zero.

$$s^{n+\frac{1}{2}}_{i+\frac{1}{2},j,L} = \hat{s}^{n+\frac{1}{2}}_{i+\frac{1}{2},j,L} - \frac{\Delta t}{2\phi\Delta x}\left(F_{x:i+\frac{3}{2},j+\frac{1}{2}} - F_{x:i+\frac{1}{2},j+\frac{1}{2}}\right) \quad (35)$$

$$s^{n+\frac{1}{2}}_{i+\frac{1}{2},j,R} = \hat{s}^{n+\frac{1}{2}}_{i+\frac{1}{2},j,R} - \frac{\Delta t}{2\phi\Delta x}\left(F_{x:i+\frac{3}{2},j+\frac{1}{2}} - F_{x:i+\frac{1}{2},j+\frac{1}{2}}\right) \quad (36)$$

$$s^{n+\frac{1}{2}}_{i,j+\frac{1}{2},L} = \hat{s}^{n+\frac{1}{2}}_{i,j+\frac{1}{2},L} - \frac{\Delta t}{2\phi\Delta y}\left(F_{y:i+\frac{1}{2},j+\frac{3}{2}} - F_{y:i+\frac{1}{2},j+\frac{1}{2}}\right) \quad (37)$$

$$s^{n+\frac{1}{2}}_{i,j+\frac{1}{2},R} = \hat{s}^{n+\frac{1}{2}}_{i,j+\frac{1}{2},R} - \frac{\Delta t}{2\phi\Delta y}\left(F_{y:i+\frac{1}{2},j+\frac{3}{2}} - F_{y:i+\frac{1}{2},j+\frac{1}{2}}\right) \quad (38)$$

where $\vec{F} = F_{x,y}$ is evaluated from the Riemann problem solution obtained for the normal direction. A Riemann problem is advection of a non-uniform initial condition described by a left and right state[15]. The left and right states (subscripts L and R) are the high-order solutions obtained from this step. With these values, the Riemann solver produces the proper value of the saturation at the edges.

The Riemann Problem Solver. After computing the normal and transverse predictor steps, we must obtain the left and right state of the saturation at the cell edges. In order to obtain the correct value at each edge, a Riemann problem must be solved. The solution of the Riemann problem is complicated by gravity, which makes the non-convexity effect in the flux function much stronger.

In the $\vec{x}$-direction, the problem R is nothing but an upwind of the left and right states previously calculated:

$$\mathsf{R}(u_L, u_R) = \begin{cases} \min\left|\vec{F}(u_L), \vec{F}(u_R)\right| & if\ u_L < u_R \\ \max\left|\vec{F}(u_L), \vec{F}(u_R)\right| & otherwise \end{cases} \quad (39)$$

However, in the $\vec{y}$-direction, the problem becomes more difficult. In this work, Oleinik's [14] approach will be used:

$$\frac{\vec{F}(u_R) - \vec{F}(u)}{u_R - u} \leq \frac{\vec{F}(u_R) - \vec{F}(u_L)}{u_R - u_L}\ \forall u \subseteq [u_L, u_R] \quad (40)$$

If $\{u_C\}$ are the roots of $\partial F(u_C)/\partial s = 0$ and $\min(u_L, u_R) < u_C < \max(u_L, u_R)$, then solution to the Riemann problem is:

$$\mathsf{R}(u_L, u_R, u_C) = \begin{cases} \min\left|\vec{F}(u_L), \vec{F}(u_R), \vec{F}(u_C)\right| & if\ u_L < u_R \\ \max\left|\vec{F}(u_L), \vec{F}(u_R), \vec{F}(u_C)\right| & otherwise \end{cases} \quad (41)$$

Crank-Nicolson Implicit Saturation Solver. Once we have completed the explicit part of our algorithm, we want to advance it in time with an implicit Crank-Nicolson scheme. This scheme has the advantage of being unconditionally stable and does not attenuate the high wave numbers that can exist in the problem. Some mathematical and physical modifications are introduced to the advection-diffusion equation in order to linearize our system.

Let us begin with the discretization of the saturation equation using a Crank-Nicolson scheme:

$$\phi\frac{s^{n+1}-s^{n}}{\Delta t}+\left(\vec{\nabla}\cdot\vec{F}\right)^{n+\frac{1}{2}}=-\frac{1}{2}\vec{\nabla}\cdot\left[W^{n+1}\vec{\nabla}P_c^{n+1}+W^{n}\vec{\nabla}P_c^{n}\right] \quad \text{.....(42)}$$

$$s^{n+1}+\vec{\nabla}\cdot\left(\frac{\Delta t}{2\phi}W^{n+1}\vec{\nabla}P_c^{n+1}\right)=s^{n}-\frac{\Delta t}{\phi}\left(\vec{\nabla}\cdot\vec{F}\right)^{n+\frac{1}{2}}-\vec{\nabla}\cdot\left(\frac{\Delta t}{2\phi}W^{n}\vec{\nabla}P_c^{n}\right) \quad \text{.....(43)}$$

Let k be the iteration number. After linearization, we get:

$$s^{n+1,k+1}-s^{n+1,k}+\vec{\nabla}\cdot\left(\frac{\Delta t}{2\phi}W^{n+1,k+1}\vec{\nabla}P_c^{n+1,k+1}\right)-\vec{\nabla}\cdot\left(\frac{\Delta t}{2\phi}W^{n+1,k}\vec{\nabla}P_c^{n+1,k}\right)$$

$$=s^{n,k}-s^{n+1,k}+\frac{\Delta t}{\phi}\left(\vec{\nabla}\cdot\vec{F}\right)^{n+\frac{1}{2}}-\vec{\nabla}\cdot\left(\frac{\Delta t}{2\phi}W^{n,k}\vec{\nabla}P_c^{n,k}\right) \quad \text{.....(44)}$$
$$-\vec{\nabla}\cdot\left(\frac{\Delta t}{2\phi}W^{n+1,k}\vec{\nabla}P_c^{n+1,k}\right)$$

Define $\xi^{n+1,k}$ as the group of terms from the previous iteration:

$$\xi^{n+1,k}=s^{n}-s^{n+1,k}-\frac{\Delta t}{\phi}\left(\vec{\nabla}\cdot\vec{F}\right)-\vec{\nabla}\cdot\left(\frac{\Delta t}{2\phi}W^{n}\vec{\nabla}P_c^{n}\right)-\vec{\nabla}\cdot\left(\frac{\Delta t}{2\phi}W^{n+1,k}\vec{\nabla}P_c^{n+1,k}\right) \quad \text{.....(45)}$$

and $\Delta s^{n+1,k+1}$ as the difference in the saturations between iterations:

$$\Delta s^{n+1,k+1}=s^{n+1,k+1}-s^{n+1,k} \quad \text{.....(46)}$$

After some mathematical manipulation we obtain:

$$\Delta s^{n+1,k+1}+\vec{\nabla}\cdot\left(\frac{\Delta t}{2\phi}\left(W^{n+1,k+1}\vec{\nabla}P_c^{n+1,k+1}-W^{n+1,k}\vec{\nabla}P_c^{n+1,k}\right)\right)=\xi^{n+1,k} \quad \text{.....(47)}$$

As we are interested in obtaining a linear expression for our saturation equation, further simplifications must be introduced.

We must do this in a way that has a physical and mathematical meaning, so that the properties of our equations are not altered. Recall that W^n is the harmonic average of the phase mobilities. Since W^n originates from the physical diffusion term on the right-hand-side of the advection-diffusion equation (10), we can lag it one iteration:

$$W^{n+1,k+1}\rightarrow W^{n+1,k} \quad \text{.....(48)}$$

We use a similar approach with the capillary pressure:

$$P_c^{n+1,k+1}-P_c^{n+1,k}=\frac{\partial P_c^{n+1,k}}{\partial s}\Delta s^{n+1,k+1} \quad \text{.....(49)}$$

With all these approximations, we can write our final expression for the implicit saturation solver:

$$\Delta s^{n+1,k+1}+\vec{\nabla}\cdot\left[\frac{\Delta t}{2\phi}W^{n+1,k}\frac{\partial P_c^{n+1,k}}{\partial s}\Delta s^{n+1,k+1}\right]=\xi^{n+1,k} \quad \text{.....(50)}$$

This equation will be solved using a fast multigrid solver.

The Pressure Solver. The pressure equation (13) is elliptic, because of the incompressibility assumption and will be solved using a fast multigrid algorithm. This can be written as:

$$\vec{\nabla}\cdot\left[\beta(s)\vec{\nabla}P_g\right]=\vec{\nabla}\cdot\left[\lambda_l\vec{\nabla}P_c-\left(\lambda_l\gamma_l+\lambda_g\gamma_g\right)\vec{\nabla}y\right] \quad \text{...(51)}$$

The Multigrid Solver. To solve the equations in our implicit steps, a fast multigrid algorithm of $\mathcal{O}(N\log N)$ operations is adopted. Multigrid successively solves residual correction equations (in this way we know the exact solution to the system of equations) using an iterative linear solver on several levels of coarser grids. There are several ways of using the coarsening, and in this work, a V-shape coarsening scheme[3] is adopted.

This algorithm is fast because it damps out high wave numbers, in contrast to other numerical solvers. We do this by moving onto successive coarser grids so that at each level the low wave numbers from the previous level become the high wave numbers in the next coarse grid, improving performance and efficiency.

We have written both of our implicit steps in the form:

$$\alpha\varphi+\vec{\nabla}\cdot\left[\beta(s)\vec{\nabla}\varphi\right]=\xi \quad \text{.....(52)}$$

which can be generalized in the form $L(\varphi)=\xi$, where L is a linear operator that satisfies $L(\varphi+\eta)=L(\varphi)+L(\eta)$ so that the same multigrid routine can be used for both the pressure and the saturation solvers.

Boundary Conditions The no-flow boundary conditions in the $\vec{x}$- and $\vec{y}$ directions are derived by imposing zero fluxes at the boundaries:

$$s_{i+1,j}=s_{i,j} \quad \text{.....(53)}$$

$$P_{i+1,j}=P_{i,j} \quad \text{.....(54)}$$

$$P_{c:i+1,j} = P_{c:i,j} \quad \text{.....(55)}$$

$$P_{p:i+1,j} = P_{p:i,j} - \rho_p g \Delta y \quad \text{.....(56)}$$

$$P_{c:i,j+1} = P_{c:i,j} - \left(\rho_g - \rho_l\right) g \Delta y \quad \text{.....(57)}$$

$$s_{c:i,j+1} = s_{c:i,j} - \frac{\left(\rho_g - \rho_l\right)}{\left.\frac{\partial P_c}{\partial s}\right|_{i,j}} g \Delta y \quad \text{.....(58)}$$

For Dirichlet boundary conditions, the values are specified at the wall and averaged to obtain the values at the ghost cells:

$$P_{p:wall} = \frac{P_{p:i,j} + P_{p:i,j+1}}{2} \quad \text{.....(59)}$$

For the multigrid solver, the non-homogeneous boundary conditions are imposed only on the finest level and the homogeneous boundary conditions are used at all coarser levels.

Numerical Results

Two stiff test problems are studied to demonstrate the strengths of the algorithm. We the algorithm's efficient memory management and robustness, even for very stiff problems.

The first problem is a 1-by-1 meter sandbox with an unstable, initially uniform, distribution of water and air saturation. The box is open at the top, and closed on the sides and the bottom. In the middle of the box, we impose an initial steep, two-dimensional "hump" of water saturation (Gaussian pulse) and let it fall by the action of gravity. The "background" uniform water also drains, while the air flows up, creating two saturation waves moving in the opposite directions. On its way down, the water "hump" is dispersed sideways by capillarity, and ultimately forms a pool at the bottom of the box.

We choose a Gaussian distribution, **Figure 2**, because it is very sharp and hard to compute on a coarse mesh. To stiffen the problem, the permeabilities are of the order of 10^{-8} m^2, with variable porosity and permeability in the y-direction.

We correlate the porosity and permeability by using a Cozeny-Karman[20] function

$$k = \frac{d_e^2 \phi^3}{180(1-\phi)^2} \quad \text{.....(60)}$$

where d_e is the pore diameter. The porosity for this case

$$\phi = 0.5 - 0.1 e^{-60(y-0.3)^2} \quad \text{.....(61)}$$

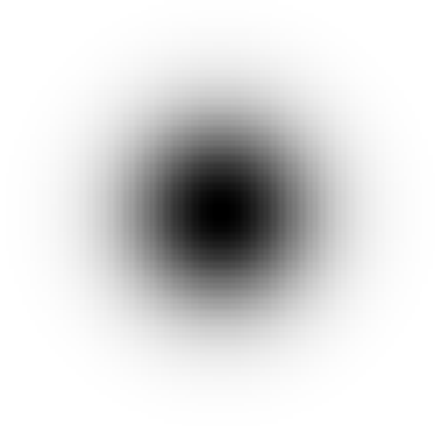

Figure 2**: The initial Gaussian pulse problem. The water saturation distribution is maximum at the center of the box.**

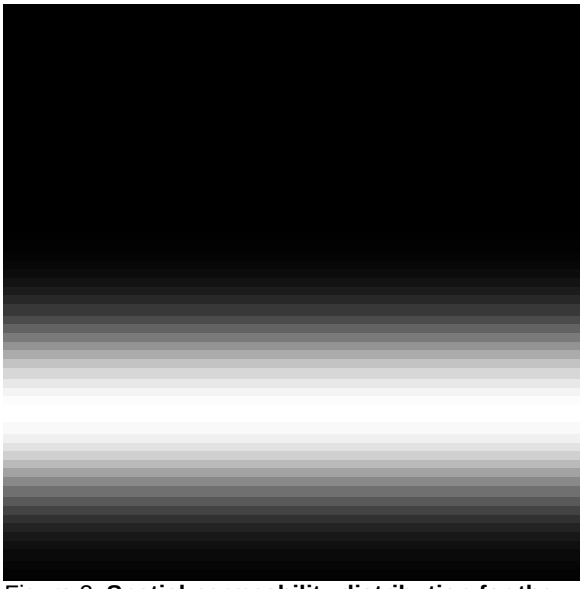

Figure 3: **Spatial permeability distribution for the falling-pulse problem. Black represents 2.5x10-^{8} m^2. White represents 8.9x 10-^{9} m^2.**

We choose this function because the quadratic exponent produces a sharp distribution of permeability and porosity, **Figure 3**.

Although this is an unusual situation for an oil reservoir, there are many cases in which we can find fast flows as in radial flow close to injectors or producers. This, again, is one of the physical situations where most of the current algorithms fail to produce a solution or the cost of producing it is very high due to the small time steps and/or large number of grid points.

Figure 4 shows the time evolution of the Gaussian pulse gravity drainage problem on a 64x64 grid. The Gaussian pulse moves very fast due to the high

permeability of the sand. It can be seen that all fronts are very sharp, which indicates the dominance of the hyperbolic components of the advection-diffusion equation. With time, and in the lower permeability region, the flow slows down and the water saturation in the pulse is spread by capillarity. At the bottom of the box, the water pools and is redirected to the sides. Finally, Figure 4 shows symmetry with respect to the vertical axis of the box. These results cannot be obtained with a classical implicit numerical simulator because the effects of numerical dispersion *would* immediately diffuse the hyperbolic components of the flow. This is what we observed when running the same case on a 64x64 grid mesh on M2Nots[19]

Some CPU times for running the gravity drainage problem are represented in **Table 1**. This results show the performance of the algorithm on a Alpha AXP running at 500 Mhz using Dec Fortran and Dec C. It can be observed from the CPU times that these scale almost linearly with the number of grid points. This fact demonstrates the high efficiency of the algorithm, specially as the number of grid points increases, which is when most of other algorithms require supercomputing tools. For the 128x128 mesh run, the maximum amount of memory required was 5028 Kbytes, fact that shows the high-efficiency of memory use in the algorithm. This mostly due to how multigrid uses and handles the arrays to be solved during the computations.

Our tests not only show the efficiency of the algorithm but also its capability to run accurately and efficiently on slow cost machines since a large number of grid blocks is not required to obtain physically meaningful solutions.

Table 1. CPU time for the Gravity Drainage Problem for t=100s

	CPU time	CPU time/number of time steps
32x32	.834	0.4305
64x64	1.666	1.330
128x128	5.18	5.455

The runs for this test problem took 3.882 min. on a 1988 IBM RS6000 PowerStation 530 with 24 MB RAM running at 33 MHz and it used a maximum of 1.2 MB of RAM (32x32 grid, 100s). Other grids were tested in this machine and the code ran without any problems. The code was also ported to a PC Clone with a AMD K6 200Mhz overclocked processor running Redhat Linux 4.2 and g77/gcc.

Figure 5. shows the time evolution of the liquid pressure distribution. Here we can see how the Gaussian pulse displaces the gas phase underneath it and how this gas escapes along the pulse sides in order to give space for the incoming water. As time passes, the system approaches a hydrostatic pressure distribution.

The main reason the algorithm does not get full second order accuracy is that the scheme is only first order accurate at the boundaries and in the neighborhood of the discontinuities. A convergence test with three grid sizes is presented on **Figure 6.**

The second test case applies directly to the oil industry. A reservoir waterflood is presented. The reservoir is 50m thick by 100 m long. A 64x64 uniform grid is used. The reservoir consists of a 5 D middle layer sandwiched between two 1 D layers. To push the limits of the algorithm, we have introduced an explicit fracture zone with a permeability of 5 D inside of the upper layer. This fracture zone is only 3 nodes wide, i.e., 2.3 m thick. The top and bottom layers are sealed. Along the left boundary, a uniform pressure of 50 atm is imposed and the right hand boundary pressure is fixed at one atm. For the water saturation boundary conditions, we use a homogeneous Neumann boundary condition, which is equivalent to a zero gradient of the capillary pressure at the injector and the producer. In order to allow water to invade the reservoir, the left vertical column of cells is filled with water. Our code does not yet handle sources and sinks for the modeling of wells, but the present setup will allow us to study our algorithm in presence of abrupt changes in permeability. It may seem exaggerated to use such a big difference in the permeabilities to model a reservoir fracture, but again, it is our purpose to demonstrate the behavior of the algorithm in the presence of discontinuities of strong changes in the model of the flow.

Figure 7 shows the numerical results. It can be observed that it takes a long time for the water to displace the oil in the reservoir. Even though the fracture is only three nodes wide, the numerical simulation models accurately the physics of the problem. The oil is expelled from the low permeability zones and into the fractures. This oil is then advected by the incoming water along the fracture. At the same time, the water in the fracture imbibes into the surrounding rock

The waterflood was simulated for 2 years. The runs took approximately 2 hours on the same IBM workstation, and with similar memory requirements.

Conclusions

- We have presented a novel, efficient, and robust algorithm to compute discontinuous incompressible, two-phase flows in two-dimensional porous media. Although this model is rather simplistic, it is currently being extended to multi-phase multicomponent transport in porous media in our current research.
- The results show that the algorithm is able to compute stiff flow problems on relatively coarse meshes and produce numerical results that resolve well the physics of the problem, even on very old and slow computers.
- Our multilevel-multigrid solvers proved to be very memory efficient even on old and slow computers.

M2Nots[19] could not run the problems in a 64x64 grid on our RS6000/530. Because our code was able to show results on the 32x32 grid (M2nots diffused the solution because anything else could happen), this shows the higher resolution capability of our code.

- Most importantly, the algorithm did not stall in the presence of permeability discontinuities, sharp fronts, fast moving fronts, and strong changes in porosity and capillary pressure. The time steps where controlled by the CFL number only in our code.
- This paper completes the first step of our research, which currently leads to the development of algorithms to compute compressible, multi-component, and multiphase non-isothermal flows.

Acknowledgments

This work is supported by the Assistant Secretary for Fossil Energy, Office of Gas and Petroleum Technology, under contract No. DE-ACO3-76FS00098 (ACTI) to the Lawrence Berkeley National Laboratory of the University of California and the Computational Science Graduate Fellowship Program of the Office of Scientific Computing in the Department of Energy. Partial funding for MA Vegas-Landeau was also provided by the Venezuelan Council for the development of sciences and technology (CONICIT). We also want to thank Dr. Kent S. Udell, Professor of Mechanical Eng for his input and comments in this work.

Nomenclature

d_e = pore diameter, m^2
dt = time step, s
F = flux function, m/s
k_r = phase relative permeability
g = gravity, m/s^2
N = number of grid points
P = phase pressure, Pa
R = Riemann solution, dimensionless
R = residual operator
s = phase saturation, dimensionless
$\tilde{s}$ = approximate phase saturation
$\hat{s}$ = intermediate value for the phase
t = time, s
u = velocity, m/s
W = harmonic average of the phase mobilities, $m^2 / Pa \cdot s$

Greek letters

β = general operator for the system of equations
ψ = slope detector function, dimensionless
Δ = difference operator, dimensionless
$\Delta^{VL}_{\lim}$ = Van Leer limiter, dimensionless
Δx = spacing in the $\vec{x}$-direction, m
Δy = spacing in the $\vec{y}$-direction, m
ϕ = porosity, dimensionless
λ = phase mobility, $m^2 / Pa \cdot s$
γ = gravity term, $kg/m^2\ s^2$
ρ = phase density, kg/m^3
σ_{lg} = liquid-gas interfacial tension, Pa-m
ξ = right-hand-side value, dimensionless
θ = Van Leer slope factor, dimensionless
Θ = contact angle, radians

Vectors

$\vec{\nabla}$ = gradient operator
$\vec{u}$ = velocity vector, m/s
$\vec{x}$ = horizontal direction vector
$\vec{y}$ = vertical direction vector

Superscripts

e = exponent of the relative permeability function
L = low
H = high
n = time level
k = iteration number
* = intermediate value

Subscripts:

c = capillary pressure
C = critical value
g = gas phase
i = index in the $\vec{x}$-direction
irr = irreducible
j = index in the $\vec{y}$-direction
l = liquid phase
L = left state
p = p^{th} phase
R = right state
y = horizontal direction
x = vertical direction
T = total velocity

References

1. Allen, M.B, Behie, G.A. and Trangenstein, J.A., *Multiphase flow in Porous Media. Mechanics, Mathematic, and Numerics*, Springer Verlag, 1988.
2. Aziz, K. and Settari, A**.,** *Petroleum Reservoir Simulation*, Applied Science, 1979.
3. Briggs, William L., *A Multigrid Tutorial*, SIAM Philadelphia, 1987
4. Colella, P., "Multidimensional Upwind Methods for Hyperbolic Conservation Laws," *Lawrence Berkeley Lab. Report LBL-17023*, 1984.
5. Colella, P., Bell, J.B. and Trangenstein, J., "Higher Order Methods for General Systems of Conservation Laws," *J. Comp. Phys*., **82**, 1989, pp. 362-397.
6. Bell, J.B, Colella, P., and Glaz, H.M., "A 2nd-order Projection Method for the incompressible Navier Stokes equations." *J. Comp. Phys*., **85**, 1989, pp. 257-283.
7. Colella, P., "A Direct Eulerian MUSCL Scheme for Gas Dynamics," *SIAM J. Sci. Stat. Comp*., **6**, 1985, pp. 104-117.
8. Colella, P., "The Piecewise-parabolic method (PPM) for Gas Dynamical Simulations," *J. Comp. Phys*., **54**, 1987, pp. 174-201.
9. Enquist B. and Osher, S., "Stable and Entropy Satisfying Approximations for Transonic Flow Calculations," *Math. Comp*., **34**, 1980, pp. 45-75.
10. Grosser, R., Carbonell, G. and Sundarasen, S., "Onset of pulsing two-phase concurrent downflow through a packed bed," *AIChE J.*, **34**, 1988, 1850.
11. Godunov, S.K., *Mat. Sb*., **47**, 1959, p.271.
12. Leveque, R.J**.**, *Numerical Methods for Conservation Laws*, Birkhauser Verlag, 1992.
13. Leverett, M.C, "Capillary behavior in porous solids," *Trans. AIME*, **34**, 1988, p. 1850.
14. Oleinik, O., "Discontinuous solutions of non-linear differential equations," *Amer. Math. Soc. Transl.* Ser. 2, **26**, 1957, pp. 95-172.
15. Osher, S. and Chakravarthy, S.,"High Resolution Schemes and the entropy condition," *SIAM J. Num. Anal*., **21**, 1984, pp. 995-984.
16. Van Leer, B., "Towards the Ultimate Conservative Difference Scheme I. The Quest of Monotonicity," *Springer Lecture Notes in Physics*., **18**, 1973, pp. 163-168.
17. Van Leer, B., "Towards the Ultimate Conservative Difference Scheme V. A Second-Order Sequel to Godunov's Method," *J. Comp. Phys*., **32**, 1979, pp. 101-136.
18. Van Leer, B., "On the Relation between Upwind-differencing Schemes of Godunov, Enquist-Osher, and Roe," *SIAM J. Sci. Stat. Comp*., **5**, 1984, pp.1-20.
19. Adenekan, A. *et al*, "Modeling of Multiphase Transport of Multicomponent Organic Contaminants and Heat in the Subsurface: Numerical Model Formulation," *Water Resources Research Journal*", 29 (11),pp. 3727-3740.

SI Metric Conversion Factors

cp	x 1.0*	E–03	= Pa. s
ft	x 3.048*	E–01	= m
md	x 9.869223	E–04	= μm^2
D	x 9.869223	E-07	= μm^2
psi	x 6.894757	E+00	= kPa

*Conversion factor is exact

SPEJ

Marco A. Vegas-Landeau, SPE, is a PhD student in the Dept. of Mechanical Engineering, U.C. Berkeley, Berkeley CA 94720, email: marco@me.berkeley.edu. He is working on sequential modeling and high order numerical schemes in multi-phase thermal flows in porous media. Vegas-Landeau holds a Mech. Engr. degree from Universidad Simon Bolivar, Caracas Venezuela and a M.S. degree from U.C. Berkeley, both in Mech. Engr.

Rick Propp, is a PhD student in the Dept. of Mechanical Engineering, U.C.-Berkeley, Berkeley CA 94720, email: propp@barkley.me.berkeley.edu. Propp is working on applied high order numerical methods for trickle bed reactors. He holds a B.S. in Mech. Engr. from Duke University.

Tadeusz W. Patzek is an Associate Professor in petroleum engineering, Department of Materials Science and Mineral Engineering., U.C.-Berkeley,Berkeley CA 94720, email: patzek@patzek.berkeley.edu. His research interests include reservoir engineering and computer assisted operations. Patzek holds a PhD in Chemical Engr. from the Sylesian Technical University, Poland.

Phillip Colella is a Professor in residence in the Mechanical. Engineering. at U.C. Berkeley, Dept. of Mechanical Engineering, U.C.-Berkeley, Berkeley CA 94720 and he is also a Senior Staff Scientist at the Lawrence Berkeley National Lab. email: pcolella@euler.berkeley.edu. His research interests include computational fluid dynamics, numerical analysis and scientific computing. Colella holds a B.S. and a PhD in Mathematics from U.C. Berkeley.

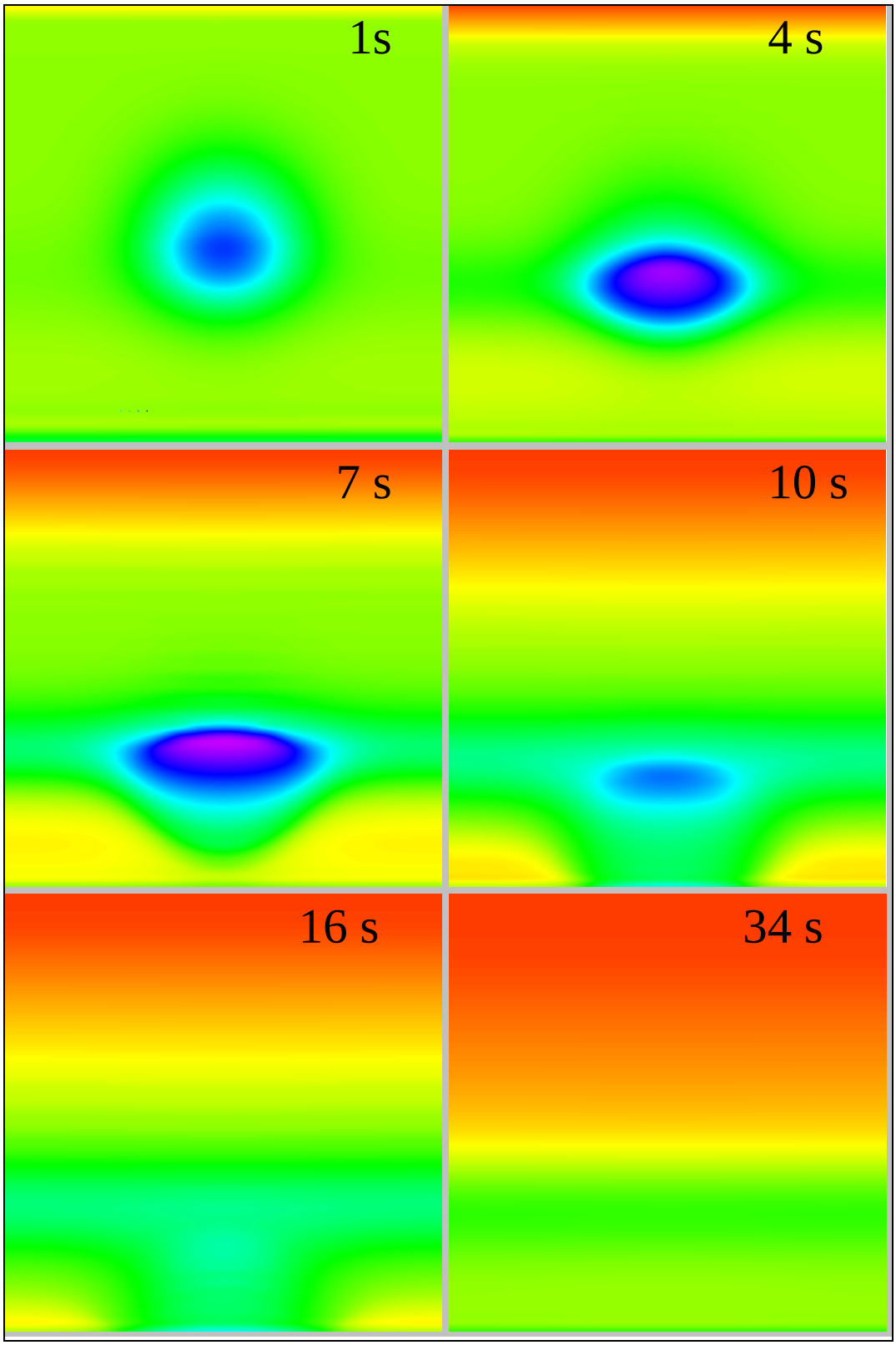

Figure 4**: Gravity drainage of a Gaussian pulse water saturation distribution in time. The plots show how due to the high permeability, the drainage behaves like a sharp moving front until the flow slows down and then it diffuses due to capillary pressure effects.**

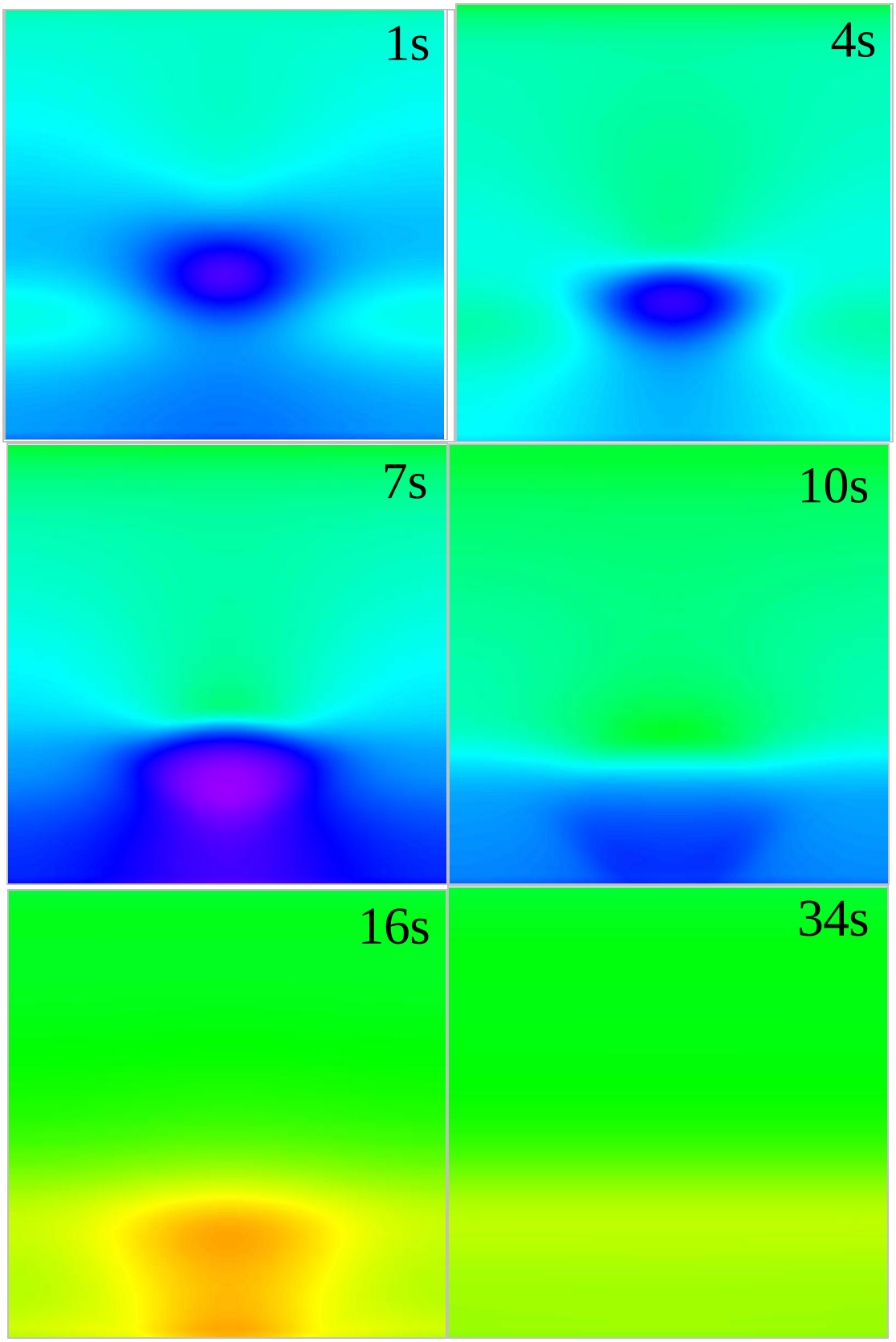

Figure 5**: Liquid pressure distribution in time for the Gaussian pulse distribution gravity-drainage problem.**

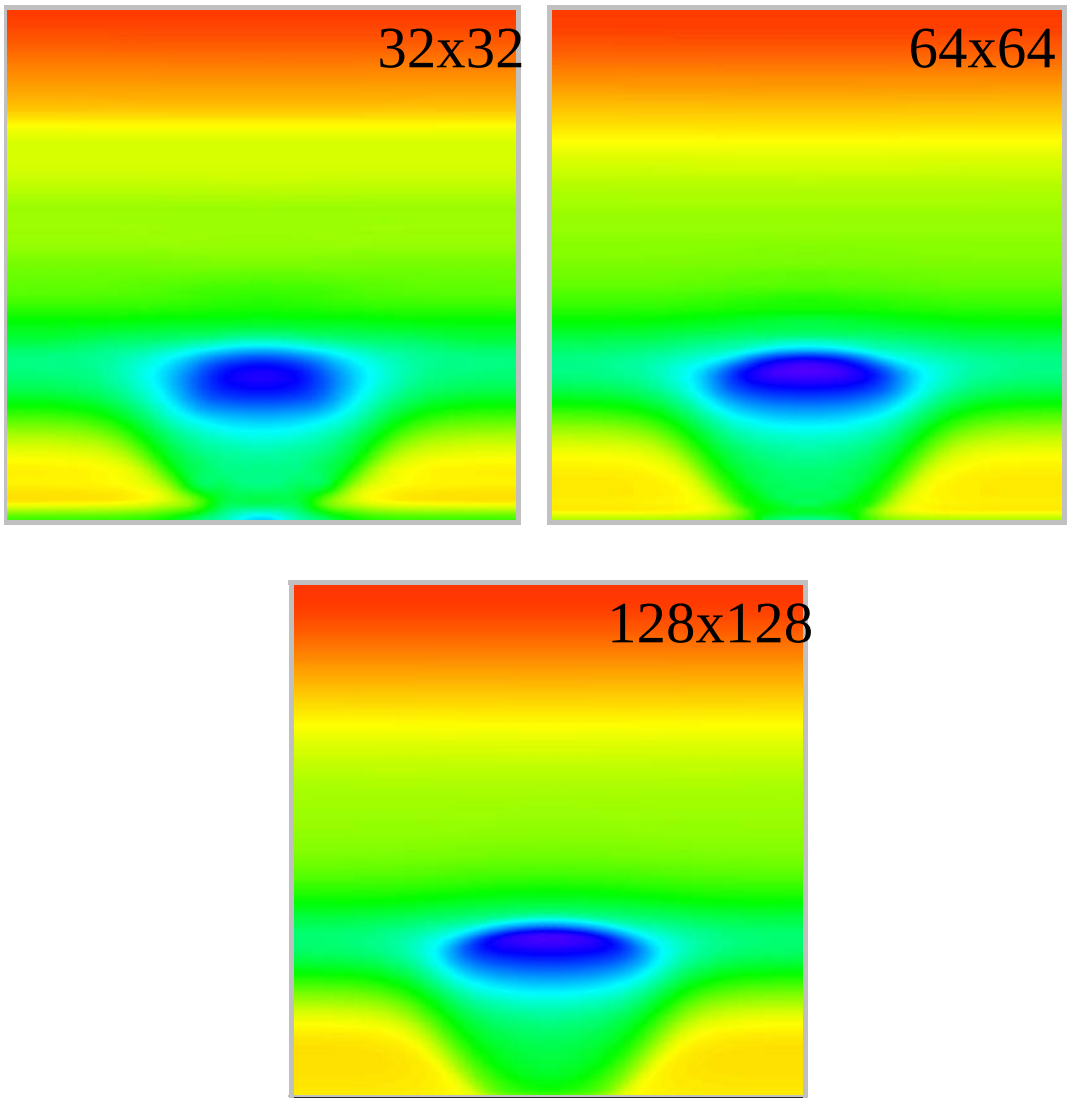

Figure 6**: Water saturation at t=100s. The results are shown on three grid sizes: 32x32, 64x64 and 128x128.**

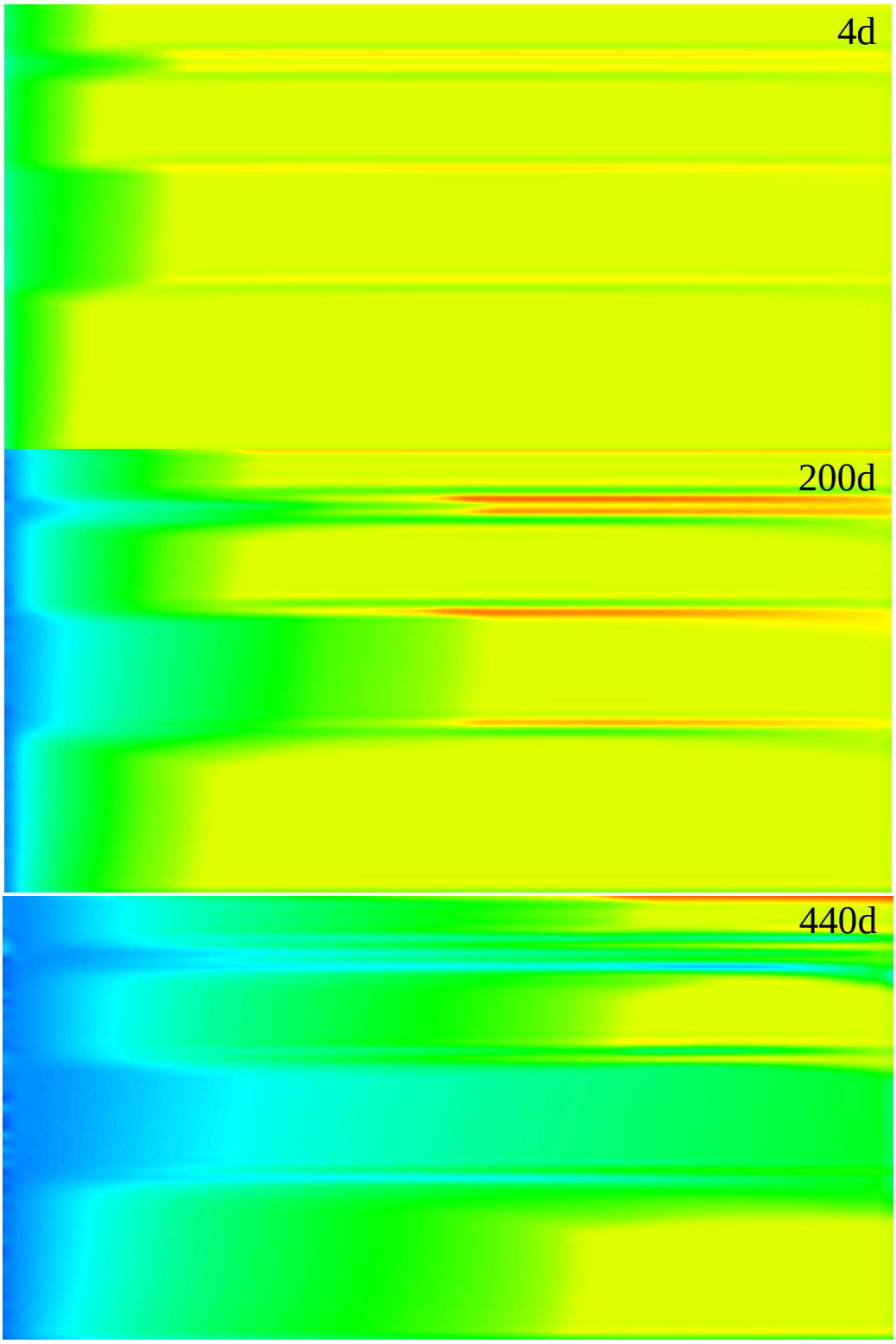

Figure 7**: Waterflood water saturation distribution in time.**