SPE INTERNATIONAL STUDENT PAPER CONTEST

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

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

This paper was prepared for presentation at the 1996 SPE Annual Meeting International Student Paper Contest.

Abstract

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 explicit-implicit approach for both accuracy and speed of the calculations. The explicit part uses a high-order Godunov scheme with Van Leer geometrical slopes, 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 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 develop a hybrid implicit-explicit scheme of a type developed by Colella *et al.*[6] for 2-D incompressible flow in porous media. The implicit-explicit scheme is intended for problems with spatially and/or temporally localized stiffness in wave speeds, such as those that can be found in radial flow near oil and gas wells, in areas with abrupt changes in permeability, such as fractures, and in phase emergence or disappearance. By stiffness, we mean that the high speed modes contain very little energy, yet determine the explicit time step through the CFL stability condition. Traditionally, such problems are not handled well by most reservoir simulators available today, as the hyperbolic components of the solution are diffused rapidly by numerical dispersion. Other simulators merely use explicit schemes at the expense of a high number of grid points, which makes them computationally expensive. Our implicit-explicit scheme has the following design principles:

1. The scheme is in a flux-conservative form.
2. The implicit-explicit hybridization is a continuous switch which is local in both time and space, i.e., depends only on the data in each computational subdomain.
3. The implicit saturation and pressure schemes satisfy a maximum principle and are unconditionally stable.
4. In the limit of steady state, the scheme is second-order accurate in space.

The sequential approach, presented by Trangenstein *et al.* [1] uses Darcy's law for the volumetric flow rates. The flow equations are manipulated mathematically to form a pressure equation and a modified equation of phase mass conservation. It has been shown that the pressure equation is elliptic and that the component-conservation equations are hyperbolic in the absence of diffusive capillarity. We have combined this approach with Colella *et al's.* [6] explicit-implicit algorithm and added gravity and diffusion terms to the system of governing equations.

A multilevel multigrid solver of the type presented in [3] is used to solve both the pressure equation and the modified component-conservation equation. This algorithm was chosen because it requires $\mathcal{O}[N\log(N)]$ number of operations, which combined with $\mathcal{O}(N)$ number of operations for the explicit schemes, gives an upper bound of $\mathcal{O}[N\log(N)]$ per time step.

Mathematical Model

Theory. In this section we will describe two-phase, two-dimensional, transient flow of an incompressible liquid (l) and gas (g).

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

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

where $\vec{u}_p$ is the phase velocity; λ_p is the phase mobility; $\gamma_p = \rho_p g$ is the specific gravity; 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:

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

subject to the usual constraint:

$$s_l + s_g = 1 \quad (3)$$

The capillary pressure is defined:

$$P_c = P_g - P_l \quad (4)$$

hence:

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

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

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

Since phases are incompressible, $\vec{\nabla}\cdot\left(\vec{u}_T\right) = 0$, i.e., the total velocity is divergence free. Substituting the Darcy velocity in the total velocity equation (6), yields:

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

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

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

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

$$\phi\frac{\partial s_l}{\partial t} + \vec{\nabla}\cdot\left[\left(\frac{\lambda_l\lambda_g}{\lambda_l+\lambda_g}\right)\left(\gamma_g - \gamma_l\right)\vec{\nabla}y + \frac{\lambda_l^2}{\lambda_l+\lambda_g}\vec{u}_T\right] = -\vec{\nabla}\cdot\left(\frac{\lambda_l\lambda_g}{\lambda_l+\lambda_g}\vec{\nabla}P_c\right) \quad (9)$$

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

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

where

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

is the flux vector, and

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

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

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

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

This is an elliptic equation that is in a form that allows easy modification to include compressible flow. Observe that 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 dimensionless capillary pressure is described by the logarithmic Leverett J-function, Grosser [10]:

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

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**. The finite difference discretization is on a rectangular grid in the horizontal direction $\vec{x}$ and the vertically upward $\vec{y}$-direction. For the time being, the number of nodes in each direction is a power of 2. 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 which are functions of saturations that are cell-centered. The question is: what is the correct saturation for evaluating the mobilities at the edges?

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.

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, but it is possible when the flow is discontinuous, because the governing differential equation no longer represents the physical problem. In this latter situation, we must use weak formulations and entropy conditions which have been extensively studied in the shock dynamics literature [15]. Only low-order schemes do not violate the Enquist-Osher conditions [23]. 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 $u_{i,j}^{n}$ as the average of $\vec{u}$ over a finite difference cell centered at $\left(i\Delta x, j\Delta y \right)$, such that:

$$u_{i,j}^{n} = \frac{1}{\Delta x \Delta y} \int_{(i-\frac{1}{2})\Delta x}^{(i+\frac{1}{2})\Delta x} \int_{(j-\frac{1}{2})\Delta y}^{(j+\frac{1}{2})\Delta y} u\left(x, y, t^{n} \right) dx dy \quad \text{.....(17)}$$

Second, time is discretized by a time step Δt, such that $t^{n+1} = t^{n} + \Delta t$. 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 $\vec{u}^{\,n+\frac{1}{2}}$ at $\frac{dt}{2}$. The corrector uses this estimate to predict the saturation and the pressure at time t^{n+1}. We do this in several steps:

1. Estimation of the phase velocities.
   a) Compute Darcy's velocities using second-order averaging for the phase mobilities.
   b) Compute total velocities (cell-centered and edge-centered).

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} and move the edge saturation obtained from the Riemann solver to the edges using a Crank-Nicolson implicit scheme.
   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 elliptic 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 can be written as:

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

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

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

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

where

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

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). We also need to compute the value of the total velocity at the center of the cell. This value can again be obtained using second-order averaging.

Fourth-Order Van Leer Slopes. The Van Leer slope limiting [22] 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, leading to a piecewise linear reconstruction $\tilde{u}_n\left(x, t^n\right)$ with a much larger total variation than that of the solution.

We can rectify this problem by applying a Van Leer slope limiter which reduces the value of the slope near the discontinuities or extreme points and it is typically designed to ensure that $TV\tilde{u}_n\left(\cdot, t_n\right) \le TVU^n$, where TV is the total variation over [0,t]. If the previous condition applies, then given solution U^n and a construction function $\tilde{u}_n\left(x, t^n\right)$, it can be shown that the algorithm is Total Variation Diminishing (TVD) for our scalar conservation law [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:

$$\Delta_{\lim} = \begin{cases} 2\min\left(\left|s_{i+1,j} - s_{i,j}\right|, \left|s_{i,j} - s_{i-1,j}\right|\right) & if\left(s_{i+1,j} - s_{i,j}\right)\left(s_{i+1,j} - s_{i,j}\right) > 0 \\ 0 & otherwise \end{cases} \quad (22)$$

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

$$\Delta^{x}_{\lim} = \min\left(\frac{2}{3}\left|s_{i+1,j} - s_{i-1,j} - \frac{1}{4}\left(\Delta^{x,f}s_{i+1,j} + \Delta^{x,f}s_{i-1,j}\right)\right|, \Delta_{\lim}\right) \quad (24)$$

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

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:

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

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. The value ε has been introduced to prevent zero-value denominators. This means that Ψ^*<0.5 is a smooth function. Ψ^*>1 indicates a moving front. Now let us introduce a safety factor:

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

where Ψ^H = 0.75, Ψ^L = 0.65, and we have introduced a safety factor of 0.15. These values are not exact and were obtained from numerical experimentation.

Let us define the following factors:

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

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

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 so that it is safe to use a high-order approximation. For $\Psi^L < \Psi < \Psi^H$, a linear interpolation of the extreme values is used.

Now that we have detected the presence of a moving front, we need to estimate the presence of the non-convex flux to complete this step. The convexity in the flux can be computed by calculating the change of sign in the derivative of the flux function:

$$\Delta^{VL}_{\lim} = \begin{cases} \theta\Delta^{VL}_{\lim} & if \quad \left.\frac{\partial F_i}{\partial s}\right|_{i-2,j} \left.\frac{\partial F_i}{\partial s}\right|_{i-1,j} < 0 \\ \theta\Delta^{VL}_{\lim} & if \quad \left.\frac{\partial F_i}{\partial s}\right|_{i-1,j} \left.\frac{\partial F_i}{\partial s}\right|_{i\ ,j} < 0 \\ \theta\Delta^{VL}_{\lim} & if \quad \left.\frac{\partial F_i}{\partial s}\right|_{i\ ,j} \left.\frac{\partial F_i}{\partial s}\right|_{i+1,j} < 0 \\ \theta\Delta^{VL}_{\lim} & if \quad \left.\frac{\partial F_i}{\partial s}\right|_{i+1,j} \left.\frac{\partial F_i}{\partial s}\right|_{i+2,j} < 0 \\ \Delta^{VL}_{\lim} & otherwise \end{cases} \quad \text{.....(30)}$$

A similar expression can be easily derived for the $\vec{y}$-direction.

The Normal and Transverse Predictors. First we derive the normal predictor step by expanding $s^n_{i,j}$ into a first-order Taylor series in time and space:

$$\hat{s}^{n+\frac{1}{2}}_{i+\frac{1}{2},j} = s^n_{i,j} + \frac{\partial s}{\partial x}\frac{\Delta x}{2} + \frac{\partial s}{\partial t}\frac{\Delta t}{2} \quad \text{.....(31)}$$

Substituting this expression in our advection-diffusion equation (10), we obtain:

$$\hat{s}^{n+\frac{1}{2}}_{i+\frac{1}{2},j} = s^n_{i,j} + \frac{\partial s}{\partial x}\frac{\Delta x}{2} - \frac{\Delta t}{2\phi}\left[\vec{\nabla}\cdot\left(W^n\vec{\nabla}P^n_c\right) + \left(\frac{\partial F_y}{\partial y}\right)^n\right] \quad \text{.....(32)}$$

where

$$\left.\frac{\partial F_x}{\partial y}\right|_{i+\frac{1}{2},j} \approx \frac{F_{y:i+\frac{1}{2},j+1} - F_{y:i+\frac{1}{2},j-1}}{\Delta y} \quad \text{.....(33)}$$

and

$$\frac{\partial s}{\partial x} \approx \frac{\Delta s^{VL}}{\Delta x} \quad \text{.....(34)}$$

Hence, we approximate the saturation slope with the high-order Van Leer slope. After combining Eqs. (31)-(34) and rearranging, we get:

$$\hat{s}^{n+\frac{1}{2}}_{i+\frac{1}{2},j} = s^n_{i,j} + \frac{1}{2}\left[1 - \frac{\Delta t}{\phi\Delta x}\max\left(0, \frac{\partial F_y}{\partial s}\right)\right]\Delta s^{VL} - \frac{\Delta t}{2\phi}\left[\vec{\nabla}\cdot\left(W^n\vec{\nabla}P^n_c\right)\frac{F_{y:i+\frac{1}{2},j+1} - F_{y:i-\frac{1}{2},j+1}}{\Delta y}\right] \quad \text{.....(35)}$$

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 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. After some manipulation, we write:

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

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

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

$$s^{n+\frac{1}{2}}_{i,j+\frac{1}{2},R} = \hat{s}^{n+\frac{1}{2}}_{i,j+\frac{1}{2},R} - \frac{\Delta t}{2\phi\Delta x}\left(F_{y:i+\frac{1}{2},j+\frac{3}{2}} - F_{y:i+\frac{1}{2},j+\frac{1}{2}}\right) \quad \text{.....(39)}$$

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 causes the non-convexity of the $\vec{F}$.

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

$$R_{mn}(u_L,u_R)=\begin{cases}\min\left|\vec{F}(u_L),\vec{F}(u_R)\right| & if\ u_L<u_R\\ \max\left|\vec{F}(u_L),\vec{F}(u_R)\right| & otherwise\end{cases} \quad .....(40)$$

However, in the $\vec{y}$-direction, the problem becomes more difficult. In this work Oleinik's [15] approach will be used:

$$\frac{\vec{F}(u_R)-\vec{F}(u)}{u_R-u}\le\frac{\vec{F}(u_R)-\vec{F}(u_L)}{u_R-u_L}\ \forall u\subseteq[u_L,u_R] \quad .....(41)$$

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

$$R_{mn}(u_L,u_R,u_C)=\begin{cases}\min\left|\vec{F}(u_L),\vec{F}(u_R),\vec{F}(u_C)\right| & if\ u_L<u_R\\ \max\left|\vec{F}(u_L),\vec{F}(u_R),\vec{F}(u_C)\right| & otherwise\end{cases}$$
$$.....(42)$$

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. This avoids the problem associated with small convergence balls of many non-linear solvers.

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

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

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

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

Define $\xi^{n+1,k}$ and $\Delta s^{n+1,k+1}$:

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

and

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

With some mathematical manipulation we obtain:

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

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:

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

We use a similar approach with the capillary pressure:

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

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

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

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. The derivation of the pressure equation is very simple:

1. Calculate $\vec{\nabla}\cdot\vec{u}_T = 0$.
2. Substitute the Darcy equation in $\vec{\nabla}\cdot\vec{u}_T = 0$.
3. Use the definition of P_c.
4. Obtain an elliptic equation that is only a function of s, P_c and P_g

After these steps, we get:

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

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 a residual correction equation (52) (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 does not damp out high wave numbers, in contrast to other numerical solvers. This is done 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 at the expense of a larger memory demand.

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

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

which can be generalized in the form $L(\varphi) = \xi$, where L is a linear operator that satisfies $L(\varphi+\eta) = L(\varphi) + L(\eta)$. Now let the F superscript denote the fine grid, and the C superscript denote the coarse grid. Let R denote the residual. The outline of the algorithm is:

1. Iterate υ_1 times on the fine grid.
2. Compute the residual on the fine grid.
3. Compute α^C, β^C and ξ^C.
4. Solve $L\left(\varphi^C\right) = \xi^C$.
5. Interpolate φ^C onto the fine grid.
6. Iterate υ_2 times on the fine grid.

For the iteration scheme, we use a Gauss-Seidel with red-black ordering because it is easy to vectorize. Also, we set υ_1 and υ_2 equal to 1 because it works well in practice. The residual on the fine grid R^F is $R^F = \xi^F - L\left(\varphi^F\right)$. The averaging operator used is a simple arithmetic average. The interpolation operator is a straight injection. For edge-centered values, we use second-order averaging. The boundaries require special attention, and they are discussed next.

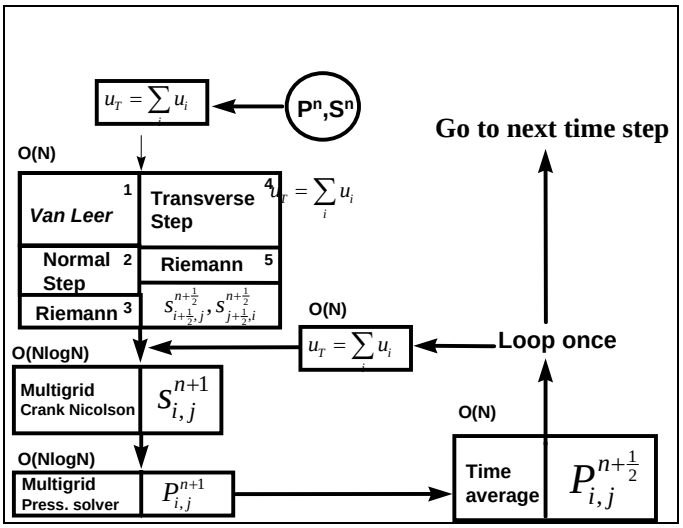

Figure 1: Outline of the High Order Sequential Explicit-Implicit Algorithm.

Boundary Conditions. Approximation of the Dirichlet and Neuman boundary conditions is second and first-order accurate, respectively.

The no-flow boundary conditions in the $\vec{x}$-direction are:

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

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

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

and in the $\vec{y}$-direction:

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

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

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

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

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

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 show that the algorithm is indeed robust and fast for very stiff problems.

The first problem is a 1-by-1 meter sand box 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 permeabilites are of the order of 10^{-8} m^2, with variable porosity and permeability in the y-direction.

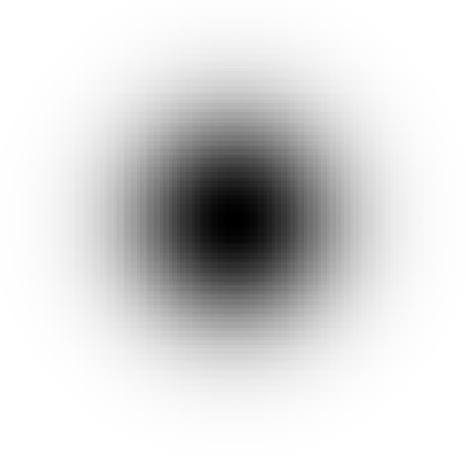

Figure 2: The initial Gaussian pulse problem. The water saturation distribution is maximum at the center of the box.

We correlate the porosity and permeability by using a Cozeny-Karaman [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

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

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

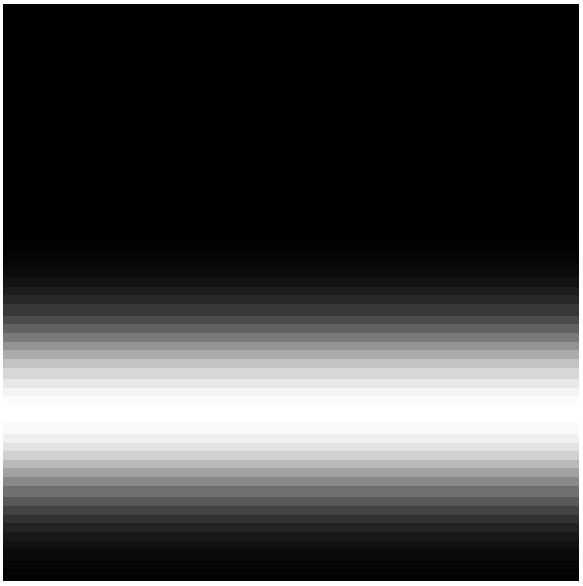

Figure 3: Spatial permeability distribution for the falling-pulse problem. Black represents 2.5x10-^{8} m^2 . White represents 8.9x 10-^{9} m^2.

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 an implicit numerical simulator because the hyperbolic components of the flow would be immediately diffused by effects of numerical dispersion.

The runs for this test problem took approximately 15 min. on an IBM RS6000 PowerStation 530 running at 33 MHz and used a maximum of 1.2 MB of RAM. This shows not only the speed of the algorithm but also its capability to run accurately

and efficiently on slow machines since a large number of grid blocks is not required to obtain physically meaningful solutions.

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.

Figure 6 shows the results for 100 s. using three sizes of grids: 32x32, 64x64 and 128x128.

The second test case applies directly to the oil industry. A reservoir waterflood is presented. The reservoir is 50m tall 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 1 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.

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.

Figure 8 shows the pressure distribution of the water phase at 60 days. Because the system is incompressible, this distribution is achieved almost immediately after the high pressure at the injector is imposed. The pressure has a linear distribution and its plane is slightly tilted to the bottom due to the effects of gravity, i.e., the pressure plane is a linear combination of the imposed pressure gradient and the hydrostatic pressure. The waterfood was simulated for 2 years. The runs took approximately 2 hours on the same IBM workstation, and with the same memory requirements.

Conclusions

- We have presented a novel, fast, and robust algorithm to compute discontinuous incompressible, two-phase flows in two-dimensional porous media.
- 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.
- Our multilevel multigrid solver proved to be very fast even on slow computers.
- 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.
- This paper completes the first step of our research, which will eventually lead to the development of algorithms to compute three-dimensional, compressible, three-phase, multicomponent, 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.

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_{mn} = 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
$\Delta_{\lim}^{VL}$ = 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 = specific gravity, 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
C = coarse
e = exponent for the relative permeability function
F = fine
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 = pth 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. Colella, P., Collins, J.P. and Glaz, H.M., "An Implicit-Explicit Eulerian Godunov Scheme for Compressible Flow," *J. Comp. Phys*., **116**, 1995, pp. 195-211.

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. Harten, A., "High Resolution Schemes for Hyperbolic Conservation Laws," *J. Comp. Phys*., **83**, 1987, pp. 357-393.

13. Lax, P.D., "Hyperbolic Systems of Conservation Laws and the Mathematical Theory of Shock Waves," *SIAM Regional Conference in Applied Mathematics*, **11**, 1972.

14. Lax, P.D., "Hyperbolic Systems of Conservation Laws," *Comm. Pure Appl. Math*., **10**, 1957, pp. 537-566.

15. Leveque, R.J**.**, *Numerical Methods for Conservation Laws*, Birkhauser Verlag, 1992.

16. Leverett,M.C, "Capillary behavior in porous solids," *Trans. AIME*, **34**, 1988, p 1850.

17. Oleinik, O., "Discontinuous solutions of non-linear differential equations," *Amer. Math. Soc. Transl.* Ser. 2, **26**, 1957, pp. 95-172.

18. Osher, S. and Chakravarthy, S.,"High Resolution Schemes and the entropy condition," *SIAM J. Num. Anal*., **21**, 1984, pp. 995-984.

19. Peaceman, D.W., *Fundamentals of Numerical Reservoir Simulation*, Elsevier Scientific Publishing Company, New York, 1977.

20. Saez, A.E. and Carbonell, R.G, "Hydrodynamic parameters for gas-liquid concurrent flow in packed beds," *AIChE J.*, **31**, 1985, p. 52.

21. Van Leer, B., "Towards the Ultimate Conservative Difference Scheme I. The Quest of Monotonicity," *Springer Lecture Notes in Physics*., **18**, 1973, pp. 163-168.

22. 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.

23. 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.

24. Woodward, P. and Colella, P., "The Numerical Simulation of Two-dimensional Fluid Flow with Strong Shocks," *J. Comp. Phys*., **54**, 1984, pp. 115-173.

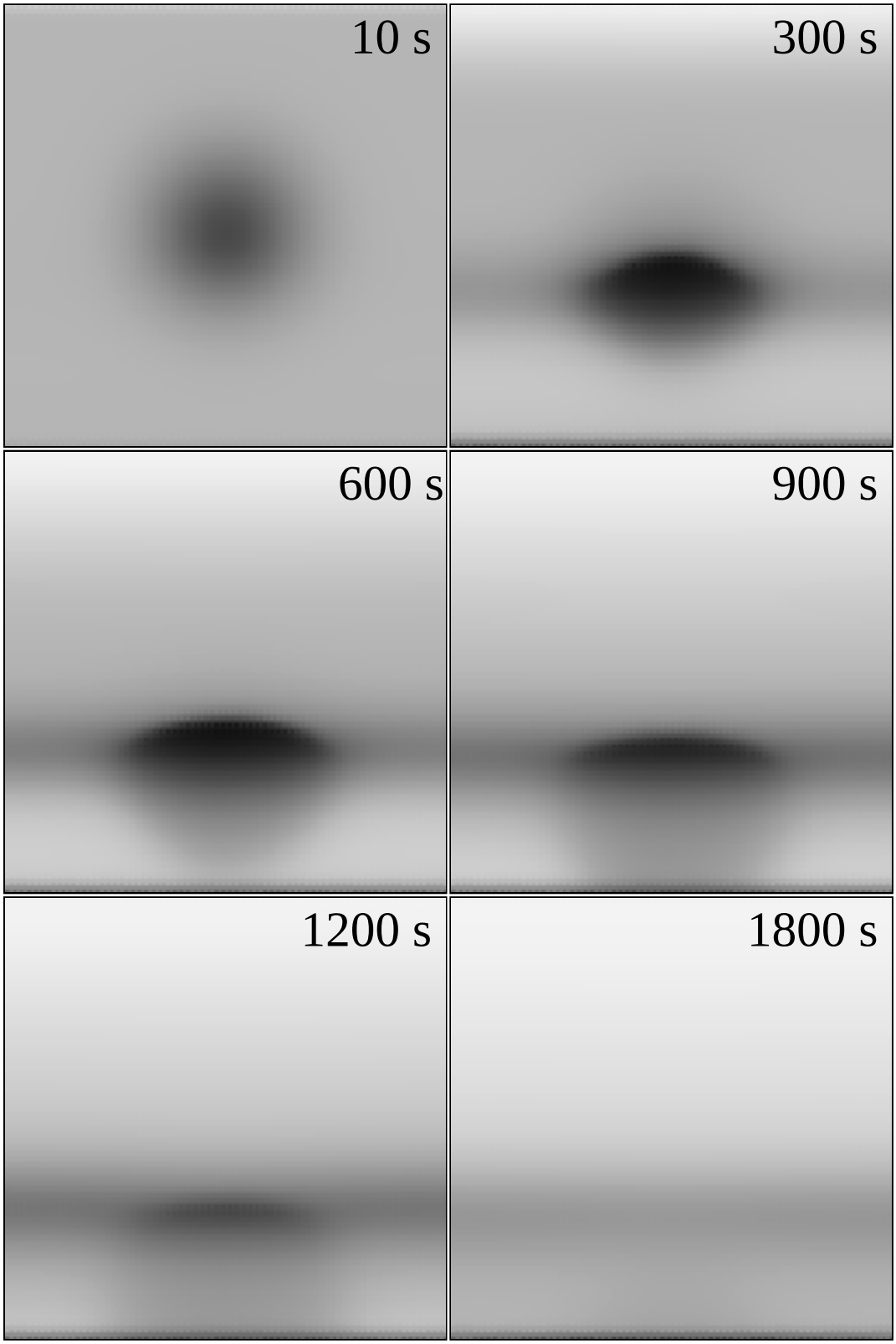

Figure 4: Gravity drainage of a Gaussian pulse water saturation distribution in time. The plots show how due to the high permeability, the drainage behaves like a sharp moving front until the flow slows down and then it diffuses due to capillary pressure effects.

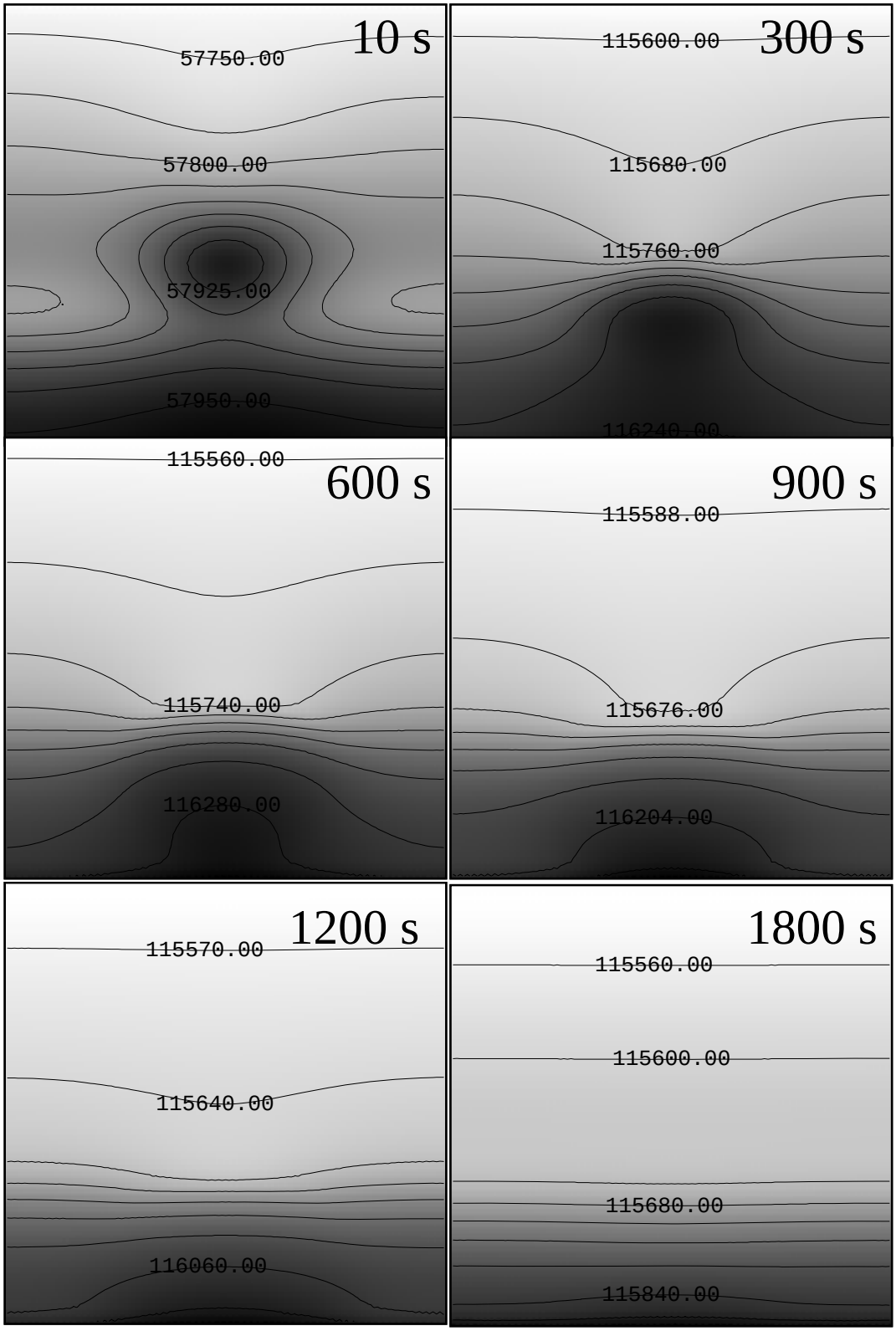

Figure 5: Liquid pressure distribution in time for the Gaussian pulse distribution gravity-drainage problem.

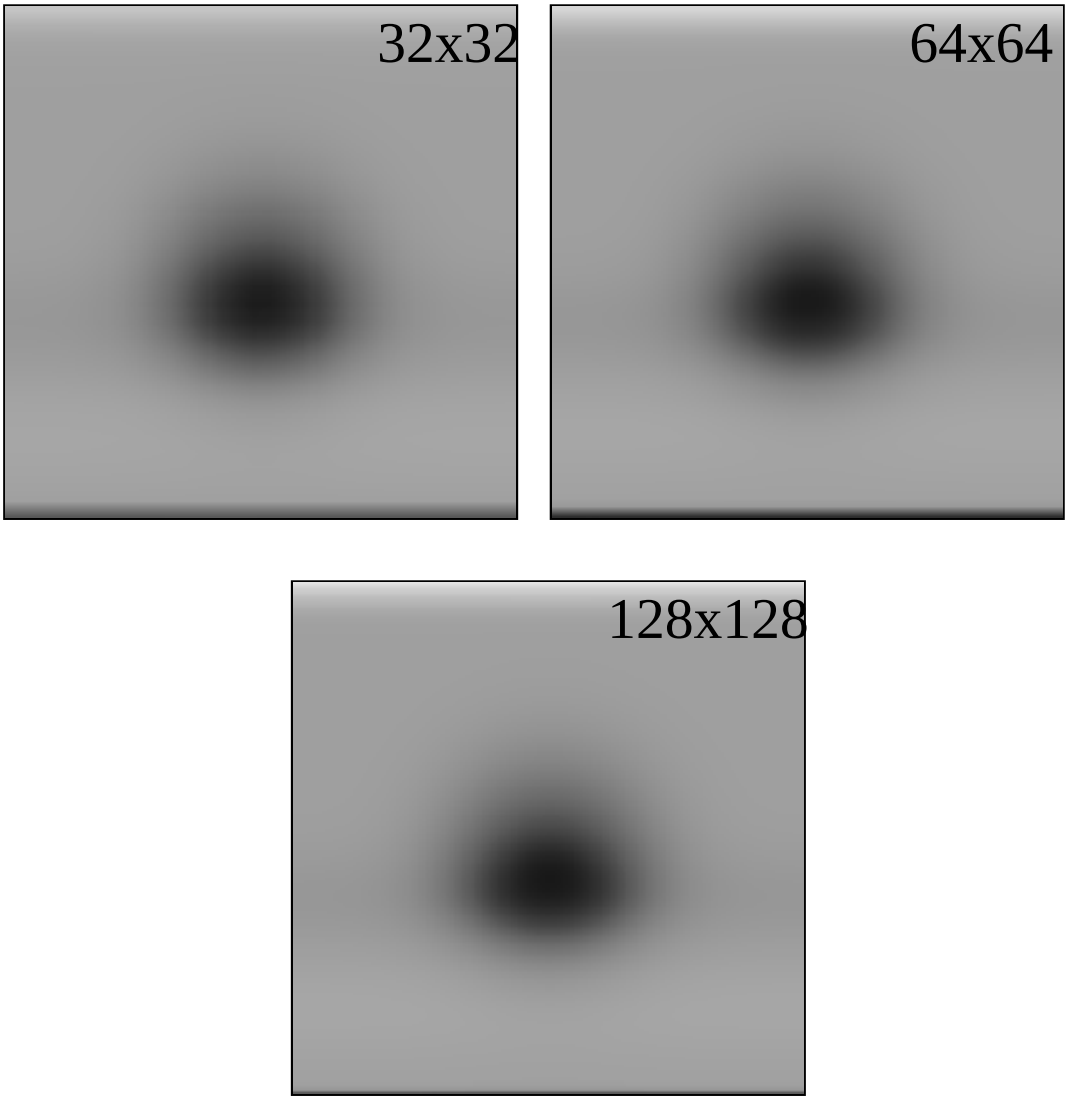

Figure 6: Water saturation at t=100s. The results are shown on three grid sizes: 32x32, 64x64 and 128x128.

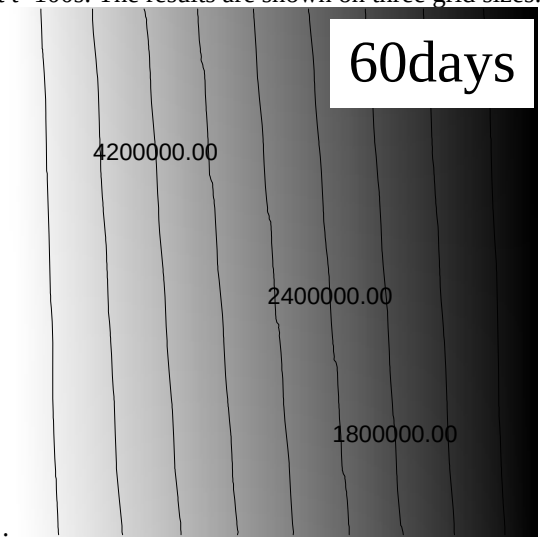

Figure 7: Pressure distribution of the waterflood. The distribution is linear, constant in time and tilted due to gravity.

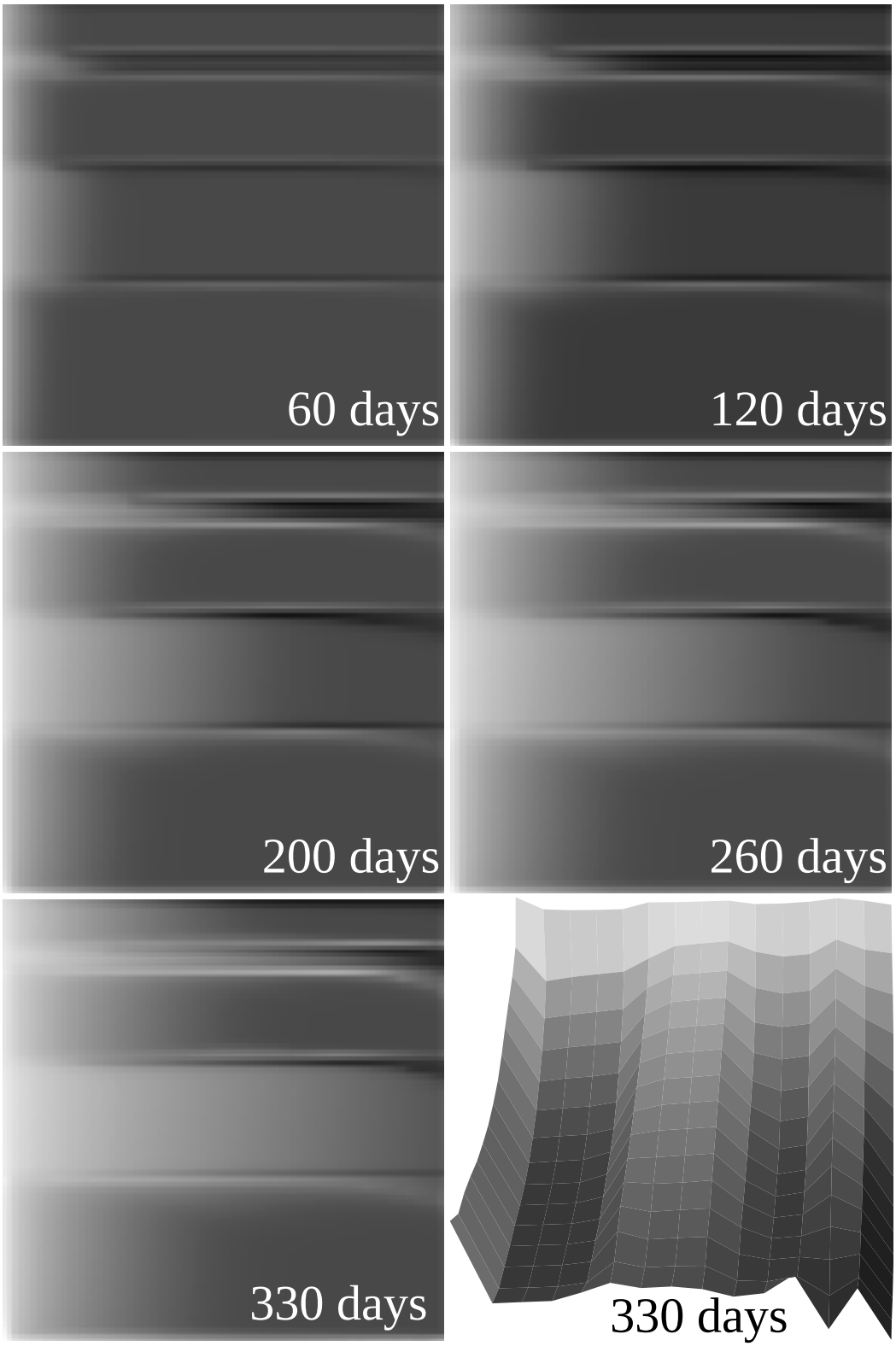

Figure 8: Waterflood water saturation distribution in time. The scale goes from white ($s_w = 1$) to black ($s_w = s_{irr}$). The last panel is a bird's eye view of higher oil saturation in the fracture and how it is being pushed by the water.