Journal of Computational Physics **165,** 311–333 (2000)
doi:10.1006/jcph.2000.6604, available online at http://www.idealibrary.com on IDEAL®

A Numerical Model for Trickle Bed Reactors[1]

Richard M. Propp, Phillip Colella, William Y. Crutchfield, and Marcus S. Day

Lawrence Berkeley National Laboratory, Berkeley, California 94720
E-mail: RMPropp@lbl.gov, PColella@lbl.gov, WYCrutchfield@lbl.gov, MSDay@lbl.gov

Received March 18, 1999; revised January 25, 2000

Trickle bed reactors are governed by equations of flow in porous media such as Darcy's law and the conservation of mass. Our numerical method for solving these equations is based on a total-velocity splitting, sequential formulation which leads to an implicit pressure equation and a semi-implicit mass conservation equation. We use high-resolution finite-difference methods to discretize these equations. Our solution scheme extends previous work in modeling porous media flows in two ways. First, we incorporate physical effects due to capillary pressure, a nonlinear inlet boundary condition, spatial porosity variations, and inertial effects on phase mobilities. In particular, capillary forces introduce a parabolic component into the recast evolution equation, and the inertial effects give rise to hyperbolic nonconvexity. Second, we introduce a modification of the slope-limiting algorithm to prevent our numerical method from producing spurious shocks. We present a numerical algorithm for accommodating these difficulties, show the algorithm is second-order accurate, and demonstrate its performance on a number of simplified problems relevant to trickle bed reactor modeling.

Key Words: trickle bed reactor; conservation laws; porous media flows; Godunov methods.

1. INTRODUCTION

A trickle bed reactor is a fixed bed of catalyst particles through which gas and liquid are allowed to flow. Typically the gas and liquid flow concurrently downward through the reactor; the liquid phase flows over the catalyst as a thin film, while the gas phase flows continuously between the catalysts [22]. These reactors have been used mainly in the

[1] Research supported at UC Berkeley by the US Department of Energy Mathematical, Information and Computational Sciences Division, Grants DE-FG03-94ER25205 and DE-FG03-92ER25140; and at the Lawrence Berkeley National Laboratory by the US Department of Energy Mathematical, Information and Computational Sciences Division, Grant DE-AC03-76SF00098. The first author was also supported by the Computational Sciences Graduate Fellowship Program of the Office of Scientific Computing in the Department of Energy.

0021-9991/00 $35.00

petroleum industry for hydrotreating processes, such as hydrodesulfurization and hydrocracking. However, there are applications in other fields, such as chemical processing, waste treatment, and biochemical processing [13].

There have been a few numerical simulations of flow distribution in a trickle bed reactor. For example, Stanek *et al.* [24] used a radial diffusion model, while Zimmerman and Ng [30] used a computer-generated model of randomly packed spheres. Anderson and Sapre [2] modeled a reactor using the same general porous media equations that are typically used in petroleum reservoir simulations (for a description of petroleum reservoir simulations, see the book by Aziz and Settari [3] or the paper by Stevenson *et al.* [25]).

Although there has been relatively little effort in modeling trickle bed reactors, similar porous media problems have been modeled extensively, mainly in the context of subsurface flows (typically petroleum reservoirs). While the same general governing equations model flow in a reactor and flow in a reservoir, there are a few key differences. First of all, there are geometry and size differences. Reactors are typically cylindrical and 10–30 meters tall [13]; on the other hand, petroleum reservoirs are typically much larger than reactors and have a large aspect ratio. Secondly, porosities are typically larger in reactors. Finally, the speed of the flow differs substantially. In a reactor, gas velocities typically range from 0.05 to 0.5 m/s, while liquid velocities range from 0.005 to 0.025 m/s. In a reservoir, velocities are typically 5×10^{-6} m/s [2].

The larger velocities in the reactor have a significant effect on the governing equations. Normally, Darcy's law is used to model the effects of phase pressure on phase velocity; however, Darcy's law is not accurate at higher velocities. To account for the discrepancies at higher velocities, we modify Darcy's law by using the Ergun equation (this is discussed in detail in Section 2.1).

The equations governing flow in a reactor are coupled and nonlinear. We reformulate these governing equations using the sequential method pioneered by Spillette [23]. In this formulation, we solve a pressure equation and then compute a total velocity; this total velocity is then used in the saturation equation instead of the individual phase velocities. Using total velocity in the saturation equation decouples the hyperbolic and elliptic pieces of the two-phase flow equations [27].

These equations are solved using the higher order Godunov techniques for hyperbolic equations outlined in Bell *et al.* [6]. Other authors have used these techniques for modeling porous media flow; for example, Trangenstein and Bell [26, 27] modeled mass transfer between phases and Nelson [18] modeled time-varying porosity.

Our work focuses on two extensions of the methods in [6]. First, we introduce more complicated physical effects into the problem and show that these effects can be modeled with second-order accuracy. We include capillary pressure effects which introduce nonlinear parabolic terms in the saturation equation. In addition, we introduce a nonlinear inlet boundary condition where we specify physically measurable quantities—the liquid velocity and the average gas velocity. We use a novel reformulation of a modified Ergun equation to model the nonlinear inertial effects due to the larger velocities present in the reactor. Second, we introduce a modification of the slope-limiting algorithm proposed by Colella [10] to prevent our numerical method from producing spurious shocks.

The rest of this paper describes the equations used to model the reactor and the numerical algorithm used to solve these equations. Section 2 describes the mathematical model of the reactor and the reformulation of the governing equations into an elliptic equation for pressure and an advection–diffusion equation for saturation. Section 3 describes the

implementation of the sequential algorithm. We discuss the solution of the pressure equation using a multigrid-accelerated iterative method [8] and the saturation equation using a Crank–Nicolson scheme and a multidimensional upwind method. Section 4 shows the effects of the slope-limiting algorithm and demonstrates that the algorithm is second-order accurate. In addition, simulations show the effects of the modified Ergun equation, capillary pressure, porosity variation, and the nonlinear inlet boundary condition.

2. MATHEMATICAL FORMULATION

In this section, we specify the governing equations, auxiliary relations, and boundary conditions for a trickle bed reactor. Then we derive a system of equations that is suitable for a sequential solution method. We assume a basic knowledge of porous media; for a more detailed examination of porous media, see Bear [4] or Collins [11].

2.1. Governing Equations

In this paper we use a simplified model of flow in the reactor. These assumptions are not necessary for our numerical algorithm to work; however, they do simplify the algorithm. We make the following assumptions about the flow:

(1) There are only two components present: component A and component B. In addition, there are only two phases: the liquid phase and the gas phase. Component A exists only in the liquid phase and component B exists only in the gas phase.
(2) The phase densities and phase viscosities are constant.
(3) Porosity is not a function of time, but it can be a function of space.

As a result of assumption (1), we can use the terms "phase" and "component" interchangeably in this paper. We will denote the liquid phase by the subscript L and the gas phase by the subscript G.

With these assumptions, the three main equations governing the flow are conservation of volume, conservation of mass, and Darcy's law,

$$s_{\mathrm{L}} + s_{\mathrm{G}} = 1 \tag{1}$$

$$\phi \frac{\partial (s_p)}{\partial t} + \nabla \cdot \mathbf{v}_p = 0 \tag{2}$$

$$\mathbf{v}_p = -\lambda_p (\nabla P_p + \gamma_p \nabla z), \tag{3}$$

where $\gamma_p = (\rho_p g / g_{\mathrm{c}})$ is a grouping of gravity terms and s_p, ρ_p, $\mathbf{v}_p$, P_p, and λ_p represent the saturation, density, velocity, pressure, and mobility of phase p. In addition, g is the acceleration due to gravity, ϕ is the porosity, g_{c} is the gravity conversion factor ($g_{\mathrm{c}} = 1$ in the metric system), t denotes time, and z is the upward-directed coordinate. These three equations must be augmented by auxiliary correlations to make the system solvable.

The phase mobility of phase p is defined as

$$\lambda_p = k_p \frac{k_{\mathrm{R}p}}{\mu_p}, \tag{4}$$

where k_p is the phase permeability, $k_{\mathrm{R}p}$ is the relative permeability of phase p, and μ_p is the viscosity of phase p. The expressions for permeability, relative permeability, and phase viscosity are typically problem dependent.

In reservoir simulations, the Kozeny–Carman equation [9] is typically used to express permeability as

$$k_p = \frac{d_e^2 \phi^3 g_c}{C_1 (1-\phi)^2}, \tag{5}$$

where d_e is the pore diameter, and $C_1 = 180$. The combination of the form of phase mobility defined in (4), the equation for permeability defined in (5), and Darcy's law (3) imply that the pressure gradient is linearly proportional to the velocity. However, several experimental and theoretical investigations have shown that there is not a simple linear dependence at larger velocities. MacDonald *et al.* ran experiments and determined that a modified Ergun equation provided a better fit to experimental data [17],

$$k_p = \left(\frac{C_1 (1-\phi)^2}{d_e^2 \phi^3 g_c} + \frac{C_2 (1-\phi) \rho_p |\mathbf{v}_p|}{\mu_p d_e \phi^3 g_c} \right)^{-1}, \tag{6}$$

where $C_1 = 180$ and $C_2 = 1.8$. We note that the first term of the equation is simply the standard permeability as expressed in the Kozeny–Carman equation (5), while the second term captures the nonlinear inertial effects. If we insert Ergun's equation (6) and the definition of phase mobility (4) into Darcy's law (3), we note that $\mathbf{v}_p$ appears on both sides of the equation; we will deal with this nonlinearity in Section 3.1.4 by reformulating Ergun's equation.

Using available experimental data on flow in packed beds, Saez and Carbonell [21] correlated the relative permeability functions as

$$k_{\mathrm{RL}} = \left(\frac{s_{\mathrm{L}} - s_{\mathrm{L_{irr}}}}{1.0 - s_{\mathrm{L_{irr}}}} \right)^{2.43} \tag{7}$$

$$k_{\mathrm{RG}} = (s_{\mathrm{G}})^{4.8}, \tag{8}$$

where $s_{\mathrm{L_{irr}}}$ is the irreducible saturation of the liquid phase. Using available data, Saez and Carbonell [21] correlated the irreducible liquid saturation as

$$s_{\mathrm{L_{irr}}} = \frac{1}{(20 + 0.9\mathrm{E\ddot{o}})\phi}, \tag{9}$$

where Eö is the Eötvos number

$$\mathrm{E\ddot{o}} = \frac{\rho_{\mathrm{L}} g d_e^2 \phi^2}{\sigma (1-\phi)^2}, \tag{10}$$

and σ is the surface tension between the gas and liquid phases.

The capillary pressure between phase L and phase G, P_{C}, is defined as

$$P_{\mathrm{C}} = P_{\mathrm{G}} - P_{\mathrm{L}}. \tag{11}$$

Leverett [16] proposed the following form for capillary pressure in an arbitrary porous media:

$$P_{\mathrm{C}} = \sigma J(s_{\mathrm{L}}) \sqrt{\frac{\phi}{k}}, \tag{12}$$

where $J(s_L)$ is a dimensionless function. Grosser *et al.* [14] approximated the J function of Leverett using

$$J(s_L) = 0.48 + 0.036 \ln\left(\frac{1 - s_L}{s_L}\right). \tag{13}$$

2.2. Total Velocity Formulation

The equations of flow in porous media exhibit both elliptic and parabolic behavior. For example, pressure effects are instantaneously felt throughout the reservoir, while saturation fronts move at a finite speed [7]. Our numerical algorithm treats these effects separately by splitting the system of governing equations into an elliptic pressure equation and a hyperbolic–parabolic saturation equation.

In a manner similar to the work of Watts [29], we define a total velocity as

$$\mathbf{v}_T \equiv \mathbf{v}_L + \mathbf{v}_G. \tag{14}$$

As a result of the simplified conservation of volume (1) and conservation of mass (2) equations, the total velocity is divergence-free. Using the definition of total velocity and Darcy's law (3), we obtain the pressure equation

$$\nabla \cdot [(\lambda_L + \lambda_G)(\nabla P_G)] = \nabla \cdot [(\lambda_G \gamma_G + \lambda_L \gamma_L)(\nabla z)] + \nabla \cdot (\lambda_L \nabla P_C). \tag{15}$$

We use the total velocity to eliminate the phase velocity from the conservation of mass equation (2)

$$\phi \frac{\partial s_L}{\partial t} + \nabla \cdot (\mathbf{F}(s_L, \mathbf{v}_T)) = -\nabla \cdot [H(s_L) \nabla P_C], \tag{16}$$

where

$$\mathbf{F}(s_L, \mathbf{v}_T) = \frac{\lambda_L(\mathbf{v}_T - \mathrm{G}\,\lambda_G)}{\lambda_L + \lambda_G}$$
$$\mathbf{G} = (\gamma_L - \gamma_G)\nabla z$$
$$H = \frac{\lambda_L \lambda_G}{\lambda_L + \lambda_G}.$$

The motivation for this substitution is that in incompressible problems with no capillary pressure, the use of total velocity splits the system into elliptic and hyperbolic pieces.

We can also expand the total velocity in terms of phase mobilities and pressures:

$$\mathbf{v}_T = -(\lambda_L + \lambda_G)\nabla P_G - (\lambda_L \rho_L + \lambda_G \rho_G) g \nabla z + \lambda_L \nabla P_C. \tag{17}$$

This reformulation has resulted in three main equations: a hyperbolic–parabolic saturation equation (16), an elliptic pressure equation (15), and an equation for total velocity (17). We can now apply appropriate numerical techniques to each type of equation.

2.3. Boundary Conditions

In order to make our system solvable, we also need to specify boundary conditions. We assume that there is an inlet at the top of the reactor and an outlet at the bottom of the reactor; the other edges are assumed to be impermeable walls.

At the impermeable walls, the normal components of the velocities are zero. From Darcy's law, it follows that the pressure gradient normal to the wall is zero at vertical walls.

At the outlet, we specify the gas pressure; this condition results from the assumption that flow exits the reactor into a region at ambient pressure. In addition, we specify that the normal derivative of capillary pressure be zero; in effect, this means that we will have no boundary layer at the outlet.

At the inlet, we use two different sets of boundary conditions. In the first set of boundary conditions, we specify quantities that are easy to implement numerically in our algorithm—liquid saturation and the pressure gradient. In the second set of boundary conditions, we specify conditions that can be measured experimentally. In this case, we specify the average gas velocity at the top of the reactor, $v_{\mathrm{G}}^{\mathrm{AVG}}$, and the liquid velocity at each cell along the top of the reactor, $\mathbf{v}_{\mathrm{L}}(x, z_{\mathrm{TOP}})$.

3. NUMERICAL ALGORITHM

In the last section, we recast the governing equations into a system of three equations: an elliptic pressure equation (15), an advection–diffusion equation for saturation (16), and an equation for total velocity (17). This section discusses the solution of those three equations.

We discretize the reactor using a finite volume discretization by covering the reactor with a mesh of grid cells. We use one of two different two-dimensional coordinate systems for these grid cells: (1) an x–z Cartesian coordinate system and (2) an r–z cylindrical coordinate system. In the Cartesian coordinate system, the grid cells are rectangular of size $\triangle x$ by $\triangle z$, and are indexed in the x-direction by i and in the z-direction by j. In the cylindrical coordinate system, the grid cells are of size $\triangle r$ by $\triangle z$, and are indexed in the r-direction by i and in the z-direction by j. We discretize in time using the index n, such that the time step $\triangle t$ is the difference between discrete times t^n and t^{n+1}.

Saturations and pressures are defined at cell centers, such that

$$s_{i,j}^n \approx s((i+0.5)\triangle x, (j+0.5)\triangle z, t^n).$$

Phase mobilities and the normal components of velocities are defined at cell edges and use half indices, such that $v_{i+1/2,j}^n$ represents the x-velocity at the "right edge" of cell (i, j) and $v_{i,j+1/2}^n$ represents the z-velocity at the "top edge" of cell (i, j). Phase mobilities are computed at cell centers and averaged to cell edges. Typically we use arithmetic averaging to compute phase mobilities; however, when the equation itself implies harmonic averaging (such as the H term in (16)), then harmonic averaging is used.

The equations are discretized using standard block-centered finite difference operators. We define the discrete gradient operator G as taking a cell-centered scalar and mapping it into an edge-centered vector field. If we consider a cell-centered scalar ϕ, then we define the x-component and z-component of the gradient field as

$$G^x(\phi)|_{i+1/2,j} = \frac{\phi_{i+1,j} - \phi_{i,j}}{\triangle x}, \quad G^z(\phi)|_{i,j+1/2} = \frac{\phi_{i,j+1} - \phi_{i,j}}{\triangle z}. \tag{18}$$

We define the discrete divergence operator, D, as taking an edge-centered vector field and mapping it into a cell-centered scalar. We define the divergence operator acting on an edge-centered vector $\mathbf{f}$ as

$$D(\mathbf{f})|_{i,j} = \frac{f^x_{i+1/2,j} - f^x_{i-1/2,j}}{\Delta x} + \frac{f^z_{i,j+1/2} - f^z_{i,j-1/2}}{\Delta z}. \tag{19}$$

These discrete operators are used to discretize all of the gradients and divergences in all of the governing equations—the pressure equation (15), the saturation equation (16), and the equation for total velocity (17). As a result, there is a mathematical consistency between the equations.

3.1. Algorithm Overview

The sequential algorithm advances the liquid saturation from time t^n to time t^{n+1} using a combination of hyperbolic–parabolic and elliptic equations. The multidimensional upwind method used to solve the hyperbolic equation was developed in [10]. This work was later extended to problems that combined hyperbolic and elliptic equations in [6] and to problems with viscous terms in [5].

We denote variables at the current time by the superscript n and variables at the new time by the superscript $n+1$. In addition, we denote temporary predicted variables at time $n+1$ by the superscript $\tilde{}$. With this notation, we can express the predictor–corrector scheme to advance the saturation from n to $n+1$ through the following steps.

Step 1. Compute the gas pressure and total velocity at the current time step. We compute the gas pressure at the current time, P^n_G, by solving the pressure equation:

$$-\nabla \cdot \left[\left(\lambda^n_G + \lambda^n_L \right) \left(\nabla P^n_G \right) \right] = -\nabla \cdot \left[\left(\lambda^n_G \gamma_G + \lambda^n_L \gamma_L \right) (\nabla z) \right] - \nabla \cdot \left(\lambda^n_L \nabla P^n_C \right).$$

Next, we compute the total velocity at the current time, $\mathbf{v}^n_T$:

$$\mathbf{v}^n_T = -\left(\lambda^n_L + \lambda^n_G \right) \nabla P^n_G - \left(\lambda^n_L \gamma_L + \lambda^n_G \gamma_G \right) \nabla z + \lambda^n_L \nabla P^n_C.$$

Step 2. Trace the liquid saturation to cell edges at the half time step. We utilize the total velocity and saturation at the current time in Godunov's method to compute liquid saturation at the half time step at cell edges, $s^{n+1/2}_{\mathrm{EDGE}}$ (this will be discussed in Section 3.1.2).

Step 3. Approximate the total velocity at the half time step. First, we predict the total velocity at the next time step, $\tilde{\mathbf{v}}_T$; then, we average it with the total velocity at the current time step, $\mathbf{v}^n_T$, in order to approximate the total velocity at the half time step, $\mathbf{v}^{n+1/2}_T$.

The first step in predicting $\tilde{\mathbf{v}}_T$ is to predict the saturation at the next time step, $\tilde{s}_L$, such that

$$\phi \frac{\tilde{s}_L - s^n_L}{\triangle t} = -\nabla \cdot \mathbf{F}\left(s^{n+1/2}_{\mathrm{EDGE}}, \mathbf{v}^n_T \right) - \frac{1}{2} \nabla \cdot \left[\tilde{H} \nabla \tilde{P}_C \right] - \frac{1}{2} \nabla \cdot \left[H^n \nabla P^n_C \right],$$

where $\mathbf{F}$ is an approximation to the flux at cell edges. The $\tilde{}$ superscript on the variables $\tilde{H}$ and $\tilde{P}_C$ indicates that they are computed using the predicted saturation $\tilde{s}$. In addition, we note that this equation's nonlinear dependence on saturation makes it difficult to solve; as a result, we actually solve an $O(\triangle t^2)$ approximation to this equation. This approximation is discussed in Section 3.1.1.

Next, we use this predicted saturation to predict the pressure at the next time step, $\tilde{P}_G$:

$$-\nabla \cdot [(\tilde{\lambda}_G + \tilde{\lambda}_L)(\nabla \tilde{P}_G)] = -\nabla \cdot [(\tilde{\lambda}_G \gamma_G + \tilde{\lambda}_L \gamma_L)(\nabla z)] - \nabla \cdot (\tilde{\lambda}_L \nabla \tilde{P}_C).$$

Again, the $\tilde{}$ superscript on the phase mobilities indicates that they are computed using the predicted saturation $\tilde{s}$. Finally, we predict the total velocity at the next time step, $\tilde{\mathbf{v}}_T$:

$$\tilde{\mathbf{v}}_T = -(\tilde{\lambda}_L + \tilde{\lambda}_G)\nabla \tilde{P}_G - (\tilde{\lambda}_L \gamma_L + \tilde{\lambda}_G \gamma_G)\nabla z + \tilde{\lambda}_L \nabla \tilde{P}_C.$$

Then, we average the predicted velocity and the current velocity to obtain the velocity at the half time step, $\mathbf{v}_T^{n+1/2}$:

$$\mathbf{v}_T^{n+1/2} = \frac{1}{2}\left(\mathbf{v}_T^n + \tilde{\mathbf{v}}_T\right).$$

Step 4. Compute the liquid saturation at the next time step. Now that we have both a saturation and a velocity at the half time step, we can compute a second-order accurate saturation at the next time step, s_L^{n+1}:

$$\phi \frac{s_L^{n+1} - s_L^n}{\triangle t} = -\nabla \cdot \mathbf{F}\left(s_{\text{EDGE}}^{n+1/2}, \mathbf{v}_T^{n+1/2}\right) - \frac{1}{2}\nabla \cdot \left[H^n \nabla P_C^n\right] - \frac{1}{2}\nabla \cdot \left[H^{n+1} \nabla P_C^{n+1}\right].$$

As in Step 3, we only solve an approximation to this equation. We will now describe specific pieces of the algorithm in greater detail.

3.1.1. *Saturation Equation*

In Section 2.2, we derived the saturation equation (16):

$$\phi \frac{\partial s_L}{\partial t} + \nabla \cdot (\mathbf{F}(s_L)) = -\nabla \cdot [H(s_L)\nabla P_C]. \tag{20}$$

Dropping the L subscript from s_L for notational simplicity and using a Crank–Nicolson discretization of (20), we obtain

$$s^{n+1} + \nabla \cdot \left[\frac{1}{2}\frac{\triangle t}{\phi} H^{n+1} \nabla P_C^{n+1}\right] = s^n - \frac{\triangle t}{\phi}(\nabla \cdot \mathbf{F})^{n+1/2} - \nabla \cdot \left[\frac{1}{2}\frac{\triangle t}{\phi} H^n \nabla P_C^n\right].$$

We use a fixed-point iteration method to linearize the discretized equation around iteration k,

$$\begin{aligned} \delta s^{k+1} + \frac{1}{2}\nabla \cdot \left[\frac{\triangle t}{\phi}\left(H^{n+1,k+1}\nabla P_C^{n+1,k+1} - H^{n+1,k}\nabla P_C^{n+1,k}\right)\right] \\ = \text{rhs}^n - \text{rhs}^{n+1,k} - \frac{\triangle t}{\phi}(\nabla \cdot \mathbf{F})^{n+1/2}, \end{aligned}$$

where

$$\begin{aligned} \delta s^{k+1} &= s^{n+1,k+1} - s^{n+1,k} \\ \text{rhs}^n &= s^n - \nabla \cdot \left[\frac{1}{2}\frac{\triangle t}{\phi} H^n \nabla P_C^n\right] \\ \text{rhs}^{n+1,k} &= s^{n+1,k} - \nabla \cdot \left[\frac{1}{2}\frac{\triangle t}{\phi} H^{n+1,k} \nabla P_C^{n+1,k}\right]. \end{aligned}$$

We lag $H^{n+1,k+1}$; i.e., we use $H^{n+1,k+1} \approx H^{n+1,k}$. In addition, we rewrite the capillary pressure difference in terms of δs^{k+1}. We obtain

$$\delta s^{k+1} + \nabla \cdot \left[\frac{1}{2} \frac{\triangle t}{\phi} H^{n+1,k} \left(\frac{\partial P_C}{\partial s} \right)^{n+1,k} \nabla \delta s^{k+1} \right] = \text{rhs}^n - \text{rhs}^{n+1,k} - \frac{\triangle t}{\phi} (\nabla \cdot \mathbf{F})^{n+1/2}. \tag{21}$$

This saturation equation is discretized using the discrete divergence operator (19) and gradient operator (18). We solve this linear equation for δs^{k+1} with multigrid-accelerated Gauss–Seidel with Red–Black ordering.

We choose the time step using the Courant–Friedrichs–Lewy (CFL) condition [12],

$$\triangle t < \sigma \frac{\triangle x}{\max(\mathbf{v})},$$

where σ is the CFL number and $\max(\mathbf{v})$ indicates the maximum wave speed in the domain. For our conservation law, the wave speed $\mathbf{v}$ is

$$\mathbf{v} = \frac{1}{\phi} \frac{\partial \mathbf{F}}{\partial s}.$$

3.1.2. *Godunov's Method*

We compute the $(\nabla \cdot \mathbf{F})^{n+1/2}$ term in the saturation advection equation using a second-order extension of Godunov's method. Godunov's method is a two-step process: (1) a Taylor series extrapolation of saturations from cell centers to cell edges at the half time step and (2) the solution of a Riemann problem using the extrapolated saturations to choose the correct edge-centered flux.

Taylor extrapolation. We denote the x-component of the total velocity by u and the z-component by v. We extrapolate saturations to cell edges at the half time step using a Taylor series extrapolation and use the saturation equation (16) to replace the temporal derivative. For example, if we extrapolate from cell (i, j) to edge $(i + \frac{1}{2}, j)$ in Cartesian coordinates, we obtain

$$\begin{aligned} s_{i+1/2,j,\mathrm{L}}^{n+1/2} &= s^n + \frac{\triangle x}{2} \frac{\partial s}{\partial x} + \frac{\triangle t}{2} \frac{\partial s}{\partial t} \\ &= s^n + \frac{\triangle x}{2} \frac{\partial s}{\partial x} - \frac{\triangle t}{2} \frac{1}{\phi} (\nabla \cdot \mathbf{F}) - \frac{\triangle t}{2} \frac{1}{\phi} (\nabla \cdot H \nabla P_C). \end{aligned}$$

By expanding $\nabla \cdot \mathbf{F}$, using the chain rule on $\mathbf{F}(\mathbf{v}_T, \phi, s)$ in the x-direction, and utilizing the velocity equation (14), we obtain

$$\begin{aligned} s_{i+1/2,\mathrm{L}}^{n+1/2} = s^n &+ \frac{1}{2} \left[1 - \frac{\triangle t}{\triangle x} \frac{1}{\phi} \max\left(0, \frac{\partial F^X}{\partial s} \right) \right] \triangle x \frac{\partial s}{\partial x} - \frac{\triangle t}{2\phi} \left[\nabla \cdot \left(\frac{\lambda_L \lambda_G}{\lambda_L + \lambda_G} \nabla P_C \right) \right] \\ &+ \frac{\triangle t}{2\phi} \left[-\frac{\partial F^X}{\partial \phi} \frac{\partial \phi}{\partial x} + \frac{\partial F^X}{\partial u} \frac{\partial v}{\partial z} - \frac{\partial F^Z}{\partial z} \right]. \end{aligned} \tag{22}$$

Since we are extrapolating from cell (i, j) to edge $(i + (1/2), j)$, we only wish to consider information (saturations, eigenvalues, etc.) from cell (i, j). We enforce this condition by

using the max term in (22); this ensures that we only use positive characteristic wave speeds. We will use (22) to extrapolate the saturations to cell edges from the left.

Since $\mathbf{F}(\mathbf{v}_T, \phi, s)$ is a known function, we can derive analytical expressions for $\partial F^X/\partial s$ and $\partial F^X/\partial u$. The computation of the phase mobilities, capillary pressure, and porosity is also straightforward.

We approximate $\partial F^X/\partial \phi$ by

$$\frac{\partial F}{\partial \phi} \approx \frac{F^X(\phi+\epsilon) - F^X(\phi-\epsilon)}{2\epsilon},$$

where $\epsilon = 0.001$. In addition, we can approximate $\partial u/\partial x$ and $\partial F^Z/\partial z$ as

$$\frac{\partial u}{\partial x} \approx \frac{u_{i+1/2,j} - u_{i-1/2,j}}{\triangle x}$$
$$\frac{\partial F^Z}{\partial z} \approx \frac{F^{Z,\mathrm{TOP}} - F^{Z,\mathrm{BOT}}}{\triangle z},$$

where $F^{Z,\mathrm{TOP}}$ and $F^{Z,\mathrm{BOT}}$ are the fluxes obtained from solving the Riemann problem at the top and bottom edges of cell (i, j). We define $F^{Z,\mathrm{TOP}}$ and $F^{Z,\mathrm{BOT}}$ as

$$F^{Z,\mathrm{TOP}} = F^{\mathrm{RP}}(s_{i,j}, s_{i,j+1})$$
$$F^{Z,\mathrm{BOT}} = F^{\mathrm{RP}}(s_{i,j-1}, s_{i,j}),$$

where F^{RP} indicates the solution to the Riemann problem. The solution to the Riemann problem will be discussed later in this section.

We approximate $\partial s/\partial x$ and $\partial \phi/\partial x$ as

$$\frac{\partial s}{\partial x} \approx \frac{\triangle s^{\mathrm{VL,M}}}{\triangle x}$$
$$\frac{\partial \phi}{\partial x} \approx \frac{\triangle \phi^{\mathrm{VL}}}{\triangle x}$$

where $\triangle\phi^{\mathrm{VL}}$ are second-order van Leer slopes [28] and $\triangle s^{\mathrm{VL,M}}$ are modified second-order van Leer slopes. van Leer slopes are defined as

$$\triangle s^{\mathrm{VL}} = \begin{cases} \mathrm{sign}(\triangle S_C)\min(2|\triangle S_L|, |\triangle S_C|, 2|\triangle S_R|), & \text{if } \triangle S_L \triangle S_R > 0 \\ 0.0, & \text{if } \triangle S_L \triangle S_R \le 0 \end{cases}$$

and the undivided differences are defined as

$$\triangle S_C = \frac{S_{i+1,j} - S_{i-1,j}}{2}$$
$$\triangle S_L = S_{i,j} - S_{i-1,j}$$
$$\triangle S_R = S_{i+1,j} - S_{i,j}.$$

Several authors [1, 6] have suggested that in the presence of non-convexity, the second-order Godunov method can produce shocks that violate the entropy condition (see Section 4.1

for an example). To prevent these entropy-violating shocks from occurring in this work, we use a variation of the approach in [6] to modify the algorithm in [10] slightly and apply it to our problem to handle regions around saturation fronts. We only modify the van Leer slopes for saturation; the porosity slope calculation remains the same.

For a given cell, we first determine if additional slope limiting is necessary. The two criteria necessary for additional limiting are (1) $\partial^2 F^X/\partial s^2$ changes sign in the region around the cell and (2) a saturation front is present. We determine the presence of a front by computing $\xi_{i,j}$ as

$$\xi_{i,j} = \frac{|s_{i+1,j} - s_{i-1,j}|}{\max(|s_{i+1,j} - s_{i-1,j}|, |s_{i+2,j} - s_{i-2,j}|, \epsilon)}.$$

We assert that a front is present in any region where $\xi > \xi^{\mathrm{H}}$ and that a smooth region occurs when $\xi < \xi^{\mathrm{L}}$. In a smooth region, we expect $\xi \approx \frac{1}{2}$; in the presence of a front, we expect $\xi \to 1$. Using these values as a guideline, we performed experiments and chose $\xi^{\mathrm{H}} = 0.75$ and $\xi^{\mathrm{L}} = 0.65$. Also based on experiments, we chose $\epsilon = 0.0001$ to prevent division by zero.

To implement the additional limiting, we compute a linearly varying smoothing function, $\tilde{\chi}_{i,j}$ as

$$\tilde{\chi}_{i,j} = \begin{cases} 1.0, & \text{if } \xi < \xi^{\mathrm{L}} \\ 1.0 - \frac{\xi - \xi^{\mathrm{L}}}{\xi^{\mathrm{H}} - \xi^{\mathrm{L}}}, & \text{if } \xi^{\mathrm{L}} < \xi < \xi^{\mathrm{H}} \\ 0.0, & \text{if } \xi > \xi^{\mathrm{H}}. \end{cases}$$

Then, we multiply the standard van Leer slopes by $\chi_{i,j}$, where

$$\chi_{i,j} = \min(\tilde{\chi}_{i-1,j}, \tilde{\chi}_{i,j}, \tilde{\chi}_{i+1,j}).$$

So, in summary, the extrapolation formula for the left edge in the Cartesian coordinate system is

$$s^{n+1/2}_{i+1/2,\mathrm{L}} = s^n + \frac{1}{2}\left[1 - \frac{\Delta t}{\Delta x}\frac{1}{\phi}\max\left(0, \frac{\partial F^X}{\partial s}\right)\right]\Delta s^{\mathrm{VL,M}} - \frac{\Delta t}{2\phi}[\nabla \cdot (H \nabla P_{\mathrm{C}})] + \frac{\Delta t}{2\phi}\left[-\frac{\partial F^X}{\partial \phi}\frac{\Delta \phi^{\mathrm{VL}}}{\Delta x} + \frac{\partial F^X}{\partial u}\frac{\Delta v}{\Delta z} - \frac{\Delta F^Z}{\Delta z}\right].$$

Extrapolation to the right edge is handled analogously, except that we will use information from cell $(i+1, j)$ instead of (i, j); as a result, we will use $\min(0, \frac{\partial F^X}{\partial s})$ instead of $\max(0, \frac{\partial F^X}{\partial s})$ to ensure that information is traveling in the correct direction.

For our purposes, the main difference between cylindrical and Cartesian coordinates is that the divergence operator looks different. Using the same techniques as in Cartesian coordinates, the extrapolation equation in the r-direction is

$$s^{n+1/2}_{i+1/2,j,\mathrm{L}} = s^n + \frac{1}{2}\left[1 - \frac{\Delta t}{\Delta r}\frac{1}{\phi}\frac{\partial F^R}{\partial s}\right]\Delta s^{\mathrm{VL,M}} - \frac{\Delta t}{2\phi}[\nabla \cdot (H \nabla P_{\mathrm{C}})] + \frac{\Delta t}{2\phi}\left[-\frac{\partial F^R}{\partial \phi}\frac{\Delta \phi^{\mathrm{VL}}}{\Delta r} - \frac{F^R}{r} - \frac{\partial F^Z}{\partial z} + \frac{\partial F^R}{\partial u}\left(\frac{\partial v}{\partial z} + \frac{u}{r}\right)\right].$$

In the z-direction, we obtain

$$s_{i,j+1/2,\mathrm{L}}^{n+1/2} = s^n + \frac{1}{2}\left[1 - \frac{\triangle t}{\triangle z}\frac{1}{\phi}\frac{\partial F^Z}{\partial s}\right]\triangle s^{\mathrm{VL,M}} - \frac{\triangle t}{2\phi}[\nabla\cdot(H\nabla P_{\mathrm{C}})] + \frac{\triangle t}{2\phi}\left[-\frac{\partial F^Z}{\partial \phi}\frac{\triangle \phi^{\mathrm{VL}}}{\triangle z} - \frac{F^R}{r} - \frac{\partial F^R}{\partial r} + \frac{\partial F^Z}{\partial v}\left(\frac{\partial u}{\partial r} + \frac{u}{r}\right)\right].$$

We compute u/r as

$$\frac{u}{r} = \frac{1}{2}\left(u_{i+1/2,\,j} + u_{i-1/2,\,j}\right)/r,$$

where r is the radius computed at the center of cell (i, j). We compute F^R/r by calculating F^R at the center of cell (i, j). In addition, we note that if the flux function is a linear function of velocity, then the extrapolation formulas in cylindrical coordinates and Cartesian coordinates are the same.

Riemann problem. The Taylor series extrapolation produces two saturations at each edge—one extrapolated from the left and one extrapolated from the right. We choose the correct edge-centered value by solving the Riemann problem. Solving the Riemann problem is central to Godunov's method; as a result, it has been widely studied in the literature. Allen *et al.* [1] and LeVeque [15] provide a general description of Riemann problems for scalar conservation laws.

Stated in general terms, the Riemann problem is the initial value problem

$$\frac{\partial s}{\partial t} + \frac{\partial F(s)}{\partial x} = 0, \tag{23}$$

with initial conditions

$$s(x, 0) = \begin{cases} s_{\mathrm{L}}, & x < 0 \\ s_{\mathrm{R}}, & x > 0. \end{cases} \tag{24}$$

Osher [20] states that the solution to this problem is to choose the s that produces a global extremum of the flux function. Specifically, if $s_{\mathrm{L}} < s_{\mathrm{R}}$, then choose the value of s between s_{L} and s_{R} that produces the minimum flux. If $s_{\mathrm{L}} > s_{\mathrm{R}}$, then choose the value of s that produces the maximum flux. This solution method requires the computation of critical states, s_{C}, such that $(\partial F/\partial s)(s_{\mathrm{C}}) = 0$. In this work, the critical states are computed using Newton's method.

3.1.3. *Pressure Equation*

In Section 2.2, we derived a pressure equation (15):

$$\nabla\cdot[(\lambda_{\mathrm{G}} + \lambda_{\mathrm{L}})(\nabla P_{\mathrm{G}})] = \nabla\cdot[(\lambda_{\mathrm{G}}\gamma_{\mathrm{G}} + \lambda_{\mathrm{L}}\gamma_{\mathrm{L}})(\nabla z)] + \nabla\cdot(\lambda_{\mathrm{L}}\nabla P_{\mathrm{C}}). \tag{25}$$

If we use the Ergun equation, then this pressure equation is nonlinear because the phase mobilities depend on the pressure. We linearize the gas pressure around iteration k,

$$\nabla\cdot\left[\left(\lambda_{\mathrm{G}}^k + \lambda_{\mathrm{L}}^k\right)\nabla\delta p^{k+1}\right] = \nabla\cdot\left[\left(\lambda_{\mathrm{G}}^k\gamma_{\mathrm{G}} + \lambda_{\mathrm{L}}^k\gamma_{\mathrm{L}}\right)\nabla z\right] + \nabla\cdot\left[\lambda_{\mathrm{L}}^k\nabla P_{\mathrm{C}} - \left(\lambda_{\mathrm{G}}^k + \lambda_{\mathrm{L}}^k\right)\nabla P_{\mathrm{G}}^k\right], \tag{26}$$

where

$$\delta p^{k+1} = P_{\mathrm{G}}^{k+1} - P_{\mathrm{G}}^{k}.$$

This pressure equation is discretized using the discrete divergence operator (19) and gradient operator (18). We solve for δp^{k+1} using multigrid-accelerated Gauss–Seidel with Red–Black ordering; each iteration is a multigrid V-cycle. Due to the nonlinearity of the problem, we must recalculate the coefficients between iterations.

3.1.4. *Darcy's Law*

In Section 2.2, we derived an expression for total velocity (17):

$$\mathbf{v}_{\mathrm{T}} = -(\lambda_{\mathrm{L}} + \lambda_{\mathrm{G}})\nabla P_{\mathrm{G}} - (\lambda_{\mathrm{L}}\rho_{\mathrm{L}} + \lambda_{\mathrm{G}}\rho_{\mathrm{G}})g\nabla z + \lambda_{\mathrm{L}}\nabla P_{\mathrm{C}}.$$

This equation is also discretized using the discrete gradient operator (18). The numerical implementation of Ergun's equation is difficult because of the nonlinearity of the phase velocity. Using the Ergun equation, Darcy's law takes the form

$$\mathbf{v}_p = -\lambda_p(s_p, |\mathbf{v}_p|, \mu_p)(\nabla P_p + \gamma_p \nabla z). \tag{27}$$

If we examine this equation, we notice that $\mathbf{v}_p$ appears on both sides of the equation; this makes solving the equation difficult as neither lagging $\mathbf{v}_p$ nor iterating on λ_p and $\mathbf{v}_p$ works very well. Lagging $\mathbf{v}_p$ effectively ignores the velocity's dependence on phase mobility which can be a significant effect. Iterating on λ_p and $\mathbf{v}_p$ frequently does not converge to a solution.

However, it is possible to manipulate equation (27) into a form where we can solve for $\mathbf{v}_p$ explicitly. Using the Kozeny–Carman correlation for permeability (5), we define A_p and B_p as

$$A_p \equiv \frac{\mu_p}{k}$$
$$B_p \equiv \left(\frac{1.8(1-\phi)\rho_p}{\mu_p d_e \phi^3 g_{\mathrm{c}}}\right)\mu_p.$$

In addition, we define a force vector, Θ_p, which includes the effects of the pressure gradient and the gravity term:

$$\Theta_p \equiv (\nabla P_p + \gamma_p \nabla z). \tag{28}$$

With these definitions, we can rewrite Darcy's law as

$$\mathbf{v}_p = -\lambda_p \Theta_p. \tag{29}$$

If we expand the phase mobility term in (29), replace all the $\mathbf{v}_p$ terms using (28), and solve the resulting quadratic equation, we obtain

$$\lambda_p(s_p, |\Theta_p|, \mu_p) = \frac{2k_{\mathrm{R}p}}{A_p\left(1 + \sqrt{1 + \frac{4B_p|\Theta_p|k_{\mathrm{R}p}}{A_p * A_p}}\right)}. \tag{30}$$

This equation expresses phase mobility in a form that is not dependent on phase velocity. If we use (30) to express the phase mobility, then the phase velocity no longer appears on both sides of (27):

$$\mathbf{v}_p = -\lambda_p(s_p, |\Theta_p|, \mu_p)(\nabla P_p + \gamma_p \nabla z). \tag{31}$$

As a result, Darcy's law is much easier to solve.

3.1.5. *State Update*

For this problem, there is a specific order in which the phase properties are computed. The first step is to compute the capillary pressure as a function of saturation using the Leverett capillary pressure correlation (12) with Grosser's correlation (13). Next, we compute the liquid pressure using the definition of capillary pressure (11). Then, we compute the phase mobilities. If we are using the Ergun equation, we must utilize the reformulation of the Ergun equation (30) to avoid the dependence of mobility on phase velocity; otherwise, we may use the more standard form of phase mobilities (4). Finally, we calculate phase velocities using Darcy's law (3).

3.1.6. *Implementation of Boundary Conditions*

The boundary conditions for this problem were specified in Section 2.3. Since the two main equations that we solve are an equation for gas pressure and an equation for liquid saturation, the specified boundary conditions must be translated into boundary conditions on gas pressure and liquid saturation.

At the impermeable walls, the physical boundary condition is that the normal component of the phase velocity is zero. Using Darcy's law, this implies

$$\frac{\partial P_{\mathrm{G}}}{\partial n} = -\rho_{\mathrm{G}} g \nabla z,$$

where n is the direction normal to the wall. For the saturation equation, we specify that the flux of saturation, F, through the wall is zero. In addition, we also specify that the saturation gradient normal to the wall is zero.

At the outlet, we specify the gas pressure and that the normal derivative of capillary pressure is zero. Assuming that the outlet is oriented in the z-direction, and using the chain rule on $P_{\mathrm{C}}(\phi, s)$, we obtain

$$\frac{\partial s}{\partial z} = -\frac{\frac{\partial P_{\mathrm{C}}}{\partial \phi}}{\frac{\partial P_{\mathrm{C}}}{\partial s}} \frac{\partial \phi}{\partial z}.$$

All three terms on the right-hand side of the equation can be calculated analytically at cell edges. If we are not including capillary pressure effects, then we specify that the saturation gradient in the z-direction is zero.

There are two cases to consider for inlet boundary conditions. In the first case, we specify the liquid saturation and gas pressure gradient. Since these are the variables we need specified, there is no need to translate the boundary conditions.

In the second case, we attempt to specify physical quantities that are easier to measure experimentally: the average gas velocity at the inlet, $v_{\mathrm{G}}^{\mathrm{AVG}}$, and the liquid velocity at each

cell along the inlet, $\mathbf{v}_{\mathrm{L}}(x, z_{\mathrm{TOP}})$. These velocity boundary conditions must be translated into boundary conditions for s_{L} and P_{G}.

We index each of the N inlet cells with the superscript i. We denote the component of the liquid velocity normal to the inlet by the scalar v_{L} and the component of the force vector (as defined in (28)) for the liquid phase that is normal to the inlet by the scalar θ_{L}. With this notation, we can write the N equations for liquid velocity as

$$v_{\mathrm{L}}^i A^i = -\lambda_{\mathrm{L}}^i \theta_{\mathrm{L}}^i A^i, \quad i = 1, N,$$

where A^i is the cross-sectional area at the inlet of cell i. In addition, we write the constraint equation for gas velocity as:

$$v_{\mathrm{G}}^{\mathrm{AVG}} * A_{\mathrm{TOP}} = -\sum_{i=1}^{N} A^i \lambda_{\mathrm{G}}^i \theta_{\mathrm{G}}^i,$$

where A_{TOP} is the cross-sectional area at the inlet for the entire reactor and θ_G^i is the component of the force vector for the gas phase that is normal to the inlet.

Using these $N + 1$ equations, we can solve for $N + 1$ unknowns at the inlet, that is, the N liquid saturations, s^i, and the gas pressure at the top, P^{TOP}. To determine these values we write the equations in residual form and then solve for s^i and P^{TOP} using a Newton iteration scheme. We repeat the Newton iterations until the sum of the squared residuals is less than our specified tolerance.

The Newton iteration equations can be written in matrix form as

$$0 = \begin{bmatrix} f_i \\ f_{N+1} \end{bmatrix} + \begin{bmatrix} \frac{\partial f_i}{\partial s^i} & \frac{\partial f_i}{\partial P^{\mathrm{TOP}}} \\ \frac{\partial f_{N+1}}{\partial s^i} & \frac{\partial f_{N+1}}{\partial P^{\mathrm{TOP}}} \end{bmatrix} \begin{bmatrix} \delta s_i \\ \delta P_{\mathrm{TOP}} \end{bmatrix},$$

where f^i is the residual for the ith equation. These residuals are defined as

$$f^i = A^i \mathbf{v}_{\mathrm{L}}^i + A^i \lambda_{\mathrm{L}}^i \theta_{\mathrm{L}}^i, \qquad i = 1, N$$

$$f^{N+1} = \sum_{i=1}^{N} \left(A^i \mathbf{v}_{\mathrm{G}}^{\mathrm{AVG}} + A^i \lambda_{\mathrm{G}}^i \theta_{\mathrm{G}}^i \right),$$

and their derivatives with respect to s^i and P^{TOP} are

$$\frac{\partial f^i}{\partial s^i} = A^i \frac{\partial \lambda_{\mathrm{L}}^i}{\partial s^i} \theta_L^i, \qquad i = 1, N$$

$$\frac{\partial f^i}{\partial P^{\mathrm{TOP}}} = A^i \left(\frac{\partial \lambda_{\mathrm{L}}^i}{\partial \theta_{\mathrm{L}}^i} \theta_{\mathrm{L}}^i + \lambda_{\mathrm{L}}^i \right) \frac{\partial \theta_{\mathrm{L}}^i}{\partial P^{\mathrm{TOP}}}, \qquad i = 1, N$$

$$\frac{\partial f^{N+1}}{\partial s^i} = A^i \frac{\partial \lambda_{\mathrm{G}}^i}{\partial s^i} \theta_{\mathrm{G}}^i, \qquad i = 1, N$$

$$\frac{\partial f^{N+1}}{\partial P^{\mathrm{TOP}}} = \sum_{i=1}^{N} A^i \left(\frac{\partial \lambda_{\mathrm{G}}^i}{\partial \theta_{\mathrm{G}}^i} \theta_{\mathrm{G}}^i + \lambda_{\mathrm{G}}^i \right) \frac{\partial \theta_{\mathrm{G}}^i}{\partial P^{\mathrm{TOP}}}.$$

This linear system can be solved to give closed-form solutions for δs^i and δP^{TOP}.

4. RESULTS

In this section, we describe the results from three series of test problems. The first test problem is a simple one-dimensional problem, for which we can construct an exact solution; here we examine the effects of the slope-limiting algorithm. The second problem is the advection of a smooth Gaussian distribution of saturation, which we use to demonstrate the second-order accuracy of the algorithm. The final problem explores the effects of capillary pressure, the Ergun equation, and variable porosity. In this section, saturation always refers to liquid saturation; the value of the gas saturation can be computed from (1).

4.1. Problem 1: Comparison with an Exact Solution

There are very few experimental or computational results in the open literature that are suitable for comparison with the model in this paper. As a result, we define a one-dimensional problem with an exact solution in order to validate the model for a simple case; in addition, we will use this simple problem to examine the effects of the slope-limiting algorithm.

In this problem, we use Cartesian coordinates and neglect the effects of gravity, capillary pressure, and the Ergun equation. We initialize the reactor as a Riemann problem centered at $x = 0.5$ with a left state of 1.0 and a right state of 0.0. We impose Dirichlet boundary conditions on saturation at $x = 0$ and $x = 1$. As a result of these boundary conditions, the total velocity is constant and the pressure equation does not need to be solved; we simply specify that the total velocity is 2.0.

The exact solution is constructed using Oleinik's solution to the Riemann problem [19]. For this problem ($s_R > s_L$), Oleinik's solution is to replace any convex portions of the flux function with straight lines. Any parts of the modified flux function that are straight lines are shocks, while any smoothly varying regions are rarefactions. Figure 1 shows the

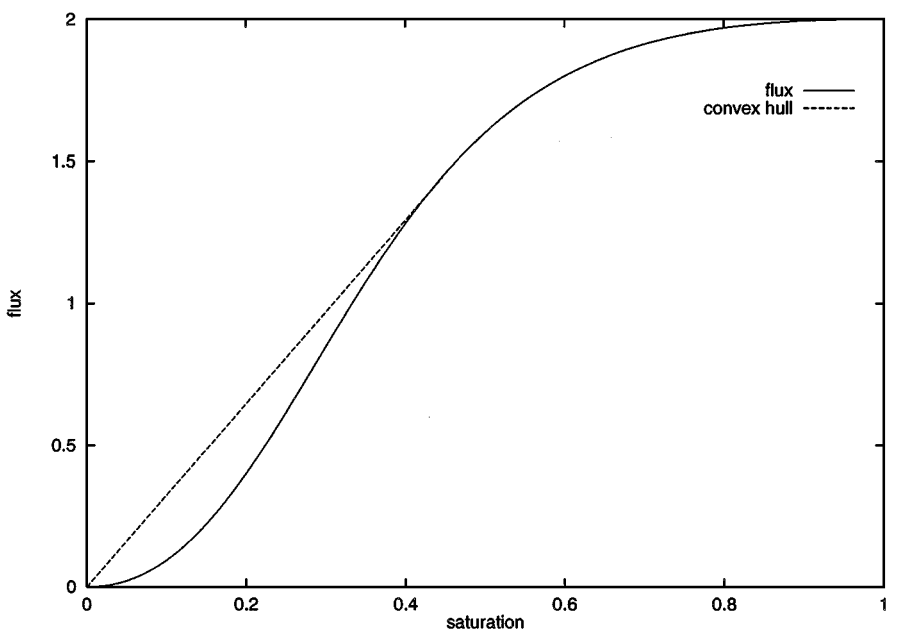

FIG. 1. Flux function and its modified convex hull. The straight line from 0 to 0.4468 indicates the presence of a shock.

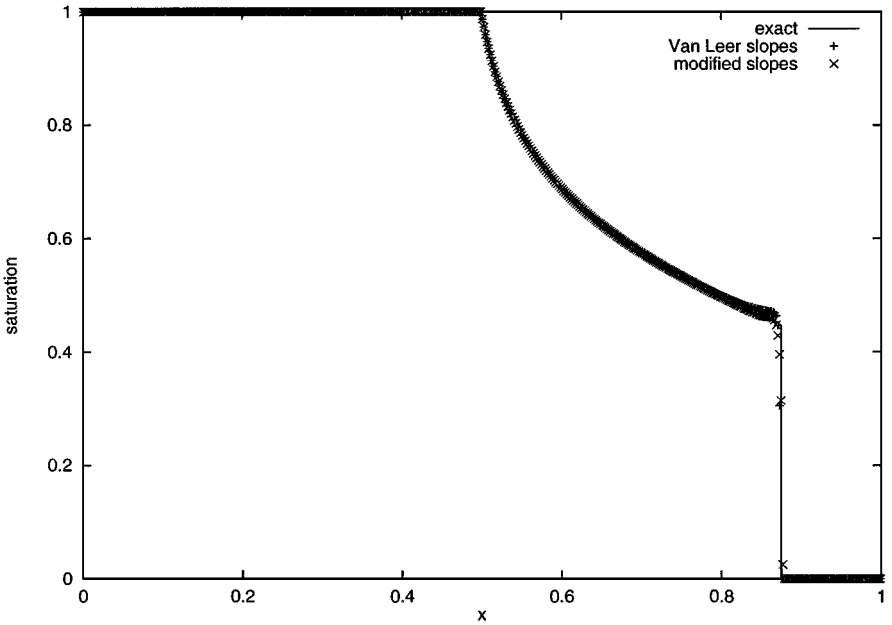

FIG. 2. Saturation profiles for the 1D problem.

flux function and its convex hull for this problem. The straight line from 0.0 to 0.4468 indicates that we expect a shock between these two saturations. In addition, we expect a rarefaction between 0.4468 and 1.0. The shock speed, in this case 3.241, follows from the Rankine–Hugoniot condition.

We run two one-dimensional simulations; one uses normal van Leer slopes, while the other uses the χ correction to the van Leer slopes. Both of these simulations are run for 350 time steps with a CFL number of 0.9 on a grid with 500 grid points. The results of these two simulations, along with the exact solution, are shown in Fig. 2. As expected, both simulations correctly predict the gross features of the exact solution.

Figure 3 shows an enlarged view of these solutions around the shock. Here we can see that the simulation using van Leer slopes predicts an incorrect location of the rarefaction and the shock (this problem is alluded to in [1, 6]). On the other hand, the simulation using modified van Leer slopes (i.e., the van Leer slopes with the χ correction) more accurately predicts the location of the rarefaction and the shock.

4.2. Problem 2: Gaussian Distribution

In this problem, we advect a Gaussian distribution of saturation in order to demonstrate the second-order accuracy of the method for smooth data. In the first case, we neglect the effects of capillary pressure and the Ergun equation, and use the Cartesian coordinate system. The reactor has a uniform porosity of 0.35. We run the simulations to the same time at four different grid resolutions—32×32, 64×64, 128×128, and 256×256.

We determined the error in the solution by comparing the saturation on a particular grid with the saturation on the next finer grid. Then, by comparing errors on successive grids, we computed convergence rates. In Table I we present the error and rates of convergence for this test problem. Due to the slope limiter in the advection algorithm, we see second-order

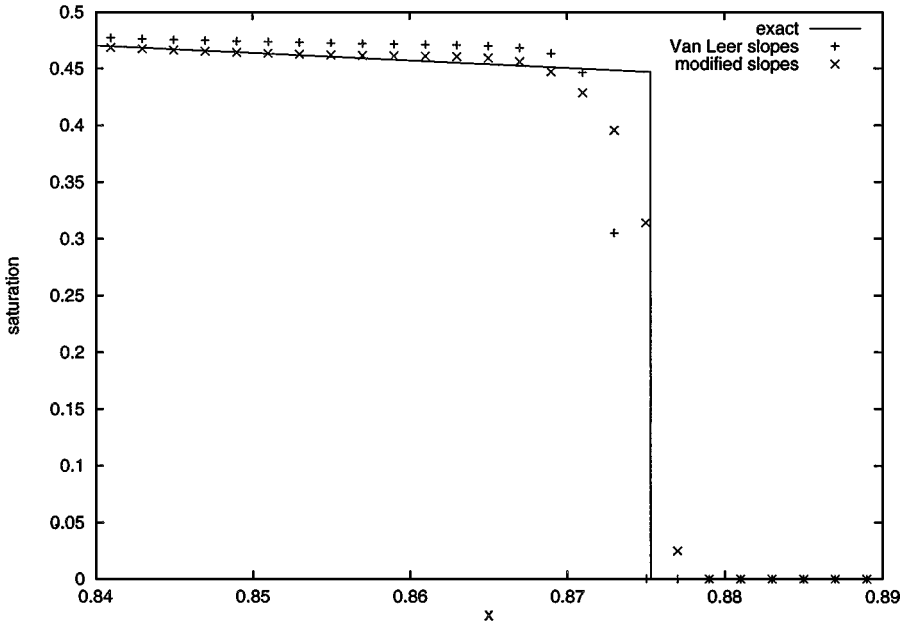

FIG. 3. A closeup of the solution for the 1D problem. The solution using normal van Leer slopes incorrectly predicts the rarefaction and the location of the shock. The solution using the χ correction, denoted as modified slopes, predicts the rarefaction and shock location more accurately.

convergence in the L_1 norm only; the L_2 and L_∞ norms are within the expected ranges for a slope-limited, second-order algorithm.

In the second case, the effects of capillary pressure and the Ergun equation are included, and the calculation is done in cylindrical coordinates. As in the first case, the reactor has a uniform porosity of 0.35 and the simulations are run to the same time on four different grid resolutions—32×32, 64×64, 128×128, and 256×256. The convergence results for these simulations are shown in Table II; again, we obtain second-order convergence for our algorithm.

4.3. Problem 3: Physical Effects

In this problem, we first demonstrate that the algorithm is second-order accurate for more complicated problems and then explore some of the more complicated physical effects present in the reactor. All of the simulations in this subsection use cylindrical coordinates and the nonlinear inlet boundary condition described in Section 3.1.6. In these simulations, we use $v_{\mathrm{G}}^{\mathrm{AVG}} = -0.01$ and $\mathbf{v}_{\mathrm{L}}(r, z_{\mathrm{TOP}}) = -0.005$. In addition, we specify that the reactor

TABLE I

Convergence Rate of Saturation in *x*–*z* Coordinates Without Capillary Pressure and Without the Ergun Equation

Norms	32–64	Rate	64–128	Rate	128–256
L_1	3.593e-5	2.05	8.662e-6	2.07	2.068e-6
L_2	1.210e-4	1.79	3.508e-5	1.79	1.011e-5
L_∞	8.957e-4	1.29	3.664e-4	1.44	1.347e-4

TABLE II

Convergence Rate of Saturation in *r–z* Coordinates with Capillary Pressure and with the Ergun Equation for Smooth Initial Conditions

Norms	32–64	Rate	64–128	Rate	128–256
L_1	1.006e-5	1.93	2.646e-6	2.01	6.548e-7
L_2	4.437e-5	1.69	1.371e-5	1.87	3.763e-6
L_∞	4.734e-4	1.23	2.018e-4	1.20	8.782e-5

TABLE III

Convergence Rate of Saturation in *r–z* Coordinates with Capillary Pressure and with the Ergun Equation

Norms	32−64	Rate	64−128	Rate	128–256
L_1	1.235e-0	1.42	1.847e-0	2.32	1.479e-0
L_2	1.048e-1	1.05	1.011e-1	1.64	6.489e-2
L_∞	1.579e-2	0.37	1.219e-2	0.93	6,399e-3

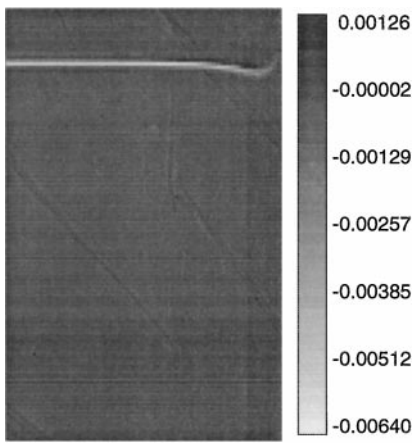

FIG. 4. Error in solution on 128 × 128 grid. The error is concentrated around the shock.

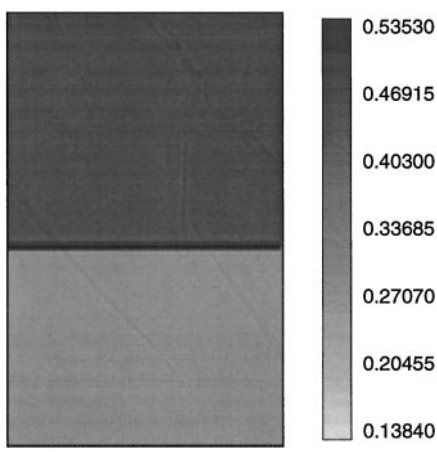

FIG. 5. Saturation at time $t = 2.0$ with no porosity variation using 12 equally spaced contours.

TABLE IV
Summary of Simulations of the Effects of Capillary Pressure and the Ergun Equation

Figure	Ergun effects	Capillary pressure effects	Porosity	Maximum saturation	Minimum saturation
5	no	no	0.35	0.46513	0.25000
6(a)	no	no	Eq. (32)	0.46848	0.25000
6(b)	yes	no	Eq. (32)	0.49578	0.25000
7(a)	no	yes	Eq. (32)	0.50618	0.13844
7(b)	yes	yes	Eq. (32)	0.53530	0.13865

has a radius of 0.14605 and a height of 0.2286; initially it contains fluid with a saturation of 0.25.

We performed a convergence study on the problem with capillary pressure, the Ergun equation, and variable porosity (this corresponds to the case shown later in Fig. 7b). The results, presented in Table III, show that even though we have added more complicated physics, we still obtain second-order convergence. Figure 4 shows that the error on the 128×128 grid is concentrated around the shock.

Next, we examine the effects of the Ergun equation, capillary pressure, and variable porosity by running five different simulations to time $t = 2.0$ on a 128×128 grid. In the first simulation, we neglect the effects of capillary pressure and the Ergun equation. In addition, the reactor has a uniform porosity of 0.35. As shown in Fig. 5, this simulation produces a one-dimensional saturation profile. In the next four simulations the porosity profile varies in the r-direction; the profile is uniform through most of the domain, but increases rapidly near the outer wall such that

$$\phi(r) = 0.35 + 0.07\left(\frac{e^{50r}}{e^{50R}}\right), \tag{32}$$

where R is the radius of the reactor.

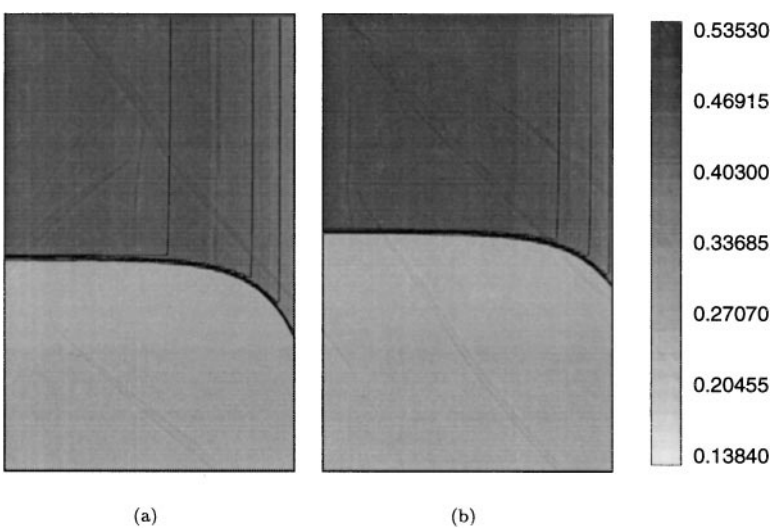

FIG. 6. Saturation with porosity variation at time $t = 2.0$ using 12 equally spaced contours: (a) without capillary pressure and without the Ergun equation; (b) without capillary pressure and with the Ergun equation.

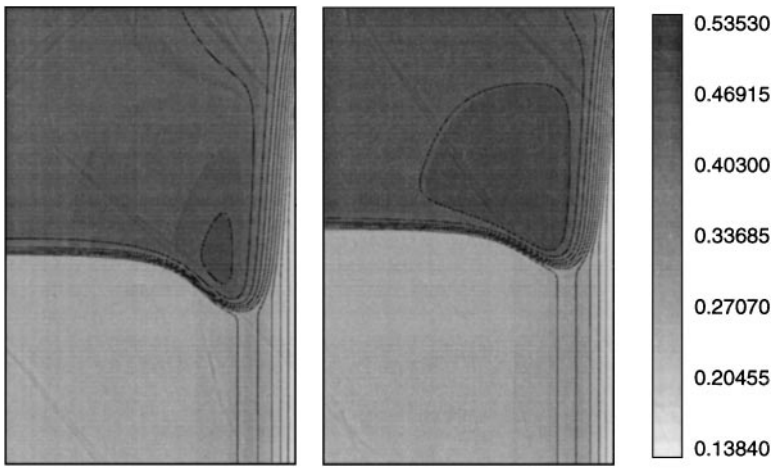

FIG. 7. Saturation with porosity variation at time $t = 2.0$ using 12 equally spaced contours: (a) with capillary pressure and without the Ergun equation; (b) with capillary pressure and with the Ergun equation.

Table IV summarizes the five simulations; in this table, "yes/no" signifies whether capillary pressure effects and Ergun effects are or are not included in the simulation. The results of these simulations are shown in Figs. 6 and 7. All of these figures use the same scale and contour interval so they can be easily compared.

By examining Figs. 6 and 7, we observe two effects of the Ergun equation: (1) it decreases the speed of the front and (2) it increases the saturation of the front. Thus, the Ergun equation appears to change the magnitude of various properties of the solution, but does not change the qualitative character of the solution.

On the other hand, capillary pressure tends to change the character of the solution. It is well known that capillary pressure acts as a diffusive force [1]; in these simulations, we note that capillary pressure causes the saturation front to "smear." In addition, in the region downstream from the front, the capillary pressure causes the liquid phase to diffuse from the high-porosity region to the low-porosity region.

5. CONCLUSIONS

This paper describes an algorithm for solving a simplified flow problem in a trickle bed reactor. Flow in a trickle bed reactor is governed by equations of flow in porous media, such as Darcy's law and conservation of mass for each component. By using a total-velocity formulation, we transformed these governing equations into a system of equations suitable for a sequential algorithm. This transformation split the problem into an elliptic pressure equation, an advection–diffusion equation that describes conservation of mass, and an equation for total velocity.

The elliptic pressure equation was solved using a multigrid-accelerated iterative method. The saturation equation was solved using a combination of Godunov's method and the multigrid-accelerated iterative method. The equation for total velocity used a reformulation of the Ergun equation that eliminated the nonlinear dependence on phase velocity. These

three equations formed the basis for the sequential predictor–corrector algorithm that we implemented.

We applied this algorithm to several test problems. For a simple one-dimensional problem, the algorithm converged to the exact solution and the modified slope limiting algorithm placed the shock and the rarefaction more accurately than standard van Leer slopes. The algorithm was shown to be second-order accurate. Finally, the code was used to explore the effects of capillary pressure, spatial porosity variations, the Ergun equation, and a nonlinear inlet boundary condition.

This work is meant as an initial step toward the design of a more complicated simulator of flow in a trickle bed reactor; several other improvements are necessary to handle the complicated physics inside a trickle bed reactor. These improvements include using a compressible gas phase, using multiple phases and components, and including heat and mass transfer and chemical reactions. Most of these changes simply involve reformulating the governing equations to include the appropriate physical effects; then the same numerical framework can be used to solve them. For example, Trangenstein and Bell [26, 17] modeled petroleum reservoir flow with multiple phases and components and a compressible gas phase by casting the governing equations in terms of component masses instead of saturations. Other potential improvements to this work are extending it to a more realistic geometry, such as a three-dimensional coordinate system, and using adaptive mesh refinement to focus the computational effort where it is needed in the problem domain.

REFERENCES

1. M. B. Allen, III, G. A. Behie, and J. A. Trangenstein, *Multiphase Flow in Porous Media*, Vol. 34 (Springer-Verlag, Berlin, 1988).
2. D. H. Anderson and A. V. Sapre, Trickle bed reactor flow simulation, *AIChE J.* **37**, 377 (1991).
3. K. Aziz and A. Settari, *Petroleum Reservoir Simulation* (Applied Science, 1979).
4. J. Bear, *Dynamics of Fluids in Porous Media* (Dover, New York, 1972).
5. J. B. Bell, P. Colella, and H. M. Glaz, An efficient second-order projection method for the incompressible Navier–Stokes equations, *J. Comput. Phys.* **85**, 257 (1989).
6. J. B. Bell, P. Colella, and J. A. Trangenstein, Higher order Godunov methods for general systems of conservation laws, *J. Comput. Phys.* **82**, 362 (1989).
7. J. B. Bell, G. R. Shubin, and J. A. Trangenstein, A method for reducing numerical dispersion in two-phase black-oil reservoir simulation, *J. Comput. Phys.* **65**, 71 (1986).
8. A. Brandt, Multi-level adaptive solutions to boundary-value problems, *Math. Comput.* **31**, 333 (1977).
9. P. C. Carman, Fluid flow through a granular bed, *Trans. Inst. Chem. Eng.* **15**, 150 (1937).
10. P. Colella, Multidimensional upwind methods for hyperbolic conservation laws, *J. Comput. Phys.* **87**, 171 (1990).
11. R. E. Collins, *Flow of Fluids through Porous Materials* (Reinhold, New York, 1961).
12. R. Courant, K. Friedrichs, and H. Lewy, On the partial difference equations of mathematical physics, *IBM J.* **11**, 215 (1967). English translation of the original work, "Uber die Partiellen Diefferenzengleichungen der Mathematischen Physik," *Math. Ann.* **100**, 32 (1928).
13. A. Gianetto and V. Specchia, Trickle-bed reactors: State of art and perspectives, *Chem. Eng. Sci.* **47**, 3197 (1992).
14. K. Grosser, R. G. Carbonell, and S. Sundaresen, Onset of pulsing in two-phase cocurrent downflow through a packed bed, *AIChE J.* **34** (1988).
15. R. J. LeVeque, *Numerical Methods for Conservation Laws*, Birkhäuser, 1990.
16. M. C. Leverett, Capillary behavior in porous solids, *Trans. AIME* **142**, 152 (1941).

17. I. F. MacDonald, M. S. El-Sayed, K. Mow, and F. A. L. Dullien, Flow through porous media—The Ergun equation revisited, *Ind. Eng. Chem. Fundam.* **18**, 199 (1979).
18. E. S. Nelson, *A Numerical Model for the Simulation of Reactive Melt Infiltration* Ph.D. thesis (University of California, Berkeley, 1998).
19. O. A. Oleinik, Discontinuous solutions of non-linear differential equations, *Am. Math. Soc. Transl.* **26**, 95 (1963).
20. S. Osher, Riemann solver, the entropy condition, and difference approximations, *SIAM J. Num. Anal.* **21**, 217 (1984).
21. A. E. Saez and R. G. Carbonell, Hydrodynamic parameters for gas-liquid cocurrent flow in packed beds, *AIChE J.* **31**, 52 (1985).
22. Y. T. Shah, *Gas–Liquid–Solid Reactor Design* (McGraw–Hill, New York, 1979).
23. A. G. Spillette, J. G. Hillestad, and H. L. Stone, A high-stability sequential solution approach to reservoir simulation, presented at the SPE 48th Annual Fall Meeting, Las Vegas, Sept. 1973.
24. V. Stanek, J. Hanika, V. Hlavacek, and O. Trnka, The effect of liquid flow distribution on the behaviour of a trickle bed reactor, *Chem. Eng. Sci.* **36**, 1045 (1981).
25. M. D. Stevenson, M. Kagan, and W. V. Pinczewski, Computational methods in petroleum reservoir simulation, *Comp. Fluids* **19**, 1 (1991).
26. J. A. Trangenstein and J. B. Bell, Mathematical structure of compositional reservoir simulation, *SIAM J. Sci. Stat. Comput.* **10**, 817 (1989).
27. J. A. Trangenstein and J. B. Bell, Mathematical structure of the black-oil model for petroleum reservoir simulation, *SIAM J. Appl. Math.* **49**, 749 (1989).
28. B. van Leer, Towards the ultimate conservative difference scheme. IV. A new approach to numerical convection, *J. Comput. Phys.* **23**, 276 (1977).
29. J. W. Watts, A compositional formulation of the pressure and saturation equations, in *7th SPE Symposium on Reservoir Simulation* (Society of Petroleum Engineers, 1983), pp. 113–122.
30. S. P. Zimmerman and K. M. Ng, Liquid distribution in trickling flow trickle-bed reactors. *Chem. Eng. Sci.* **41**, 861 (1986).