A Parallel Second-Order Adaptive Mesh Algorithm for Incompressible Flow in Porous Media

By George S. H. Pau, Ann S. Almgren, John B. Bell, Michael J. Lijewski

Lawrence Berkeley National Laboratory, Berkeley, CA 94720, USA

In this paper we present a second-order accurate adaptive algorithm for solving multiphase, incompressible flow in porous media. We assume a multiphase form of Darcy's law with relative permeabilities given as a function of the phase saturation. The remaining equations express conservation of mass for the fluid constituents. In this setting the total velocity, defined to be the sum of the phase velocities, is divergence-free. The basic integration method is based on a total-velocity splitting approach in which we solve a second-order elliptic pressure equation to obtain a total velocity. This total velocity is then used to recast component conservation equations as nonlinear hyperbolic equations. Our approach to adaptive refinement uses a nested hierarchy of logically rectangular grids with simultaneous refinement of the grids in both space and time. The integration algorithm on the grid hierarchy is a recursive procedure in which coarse grids are advanced in time, fine grids are advanced multiple steps to reach the same time as the coarse grids and the data at different levels are then synchronized. The single grid algorithm is described briefly, but the emphasis here is on the time-stepping procedure for the adaptive hierarchy. Numerical examples are presented to demonstrate the algorithm's accuracy and convergence properties and to illustrate the behavior of the method.

Keywords: adaptive mesh refinement, Darcy flow, porous media

1. Introduction

Multicomponent and multiphase flows in the subsurface are often characterized by localized phenomena such as steep concentration gradients or saturation fronts. Accurately resolving these types of phenomena requires high resolution in regions where the solution is changing rapidly. For this reason, the development of some type of dynamic gridding capability has long been of interest in the porous media community.

Heinemann (1983) and Ewing *et al.* (1989) have considered dynamic local grid refinement approaches. More recent papers by, for example, Sammon (2003) and Christensen *et al.* (2004), discuss development of adaptive techniques in the context of unstructured grids. An alternative approach to local refinement is based on structured-grid adaptive mesh refinement. This type of approach, based on the strategy introduced for gas dynamics by Berger and Colella (1989), was first applied to porous media flow by Hornung and Trangenstein (1997) and by Propp (1998). Additional developments are discussed in the work by Trangenstein (2002),

Trangenstein and Bi (2002), and Hoang and Kleppe (2006). In this approach, regions to be refined are uniformly subdivided in both space and time. Related approaches were developed by Nilsson *et al.* (2005*a*, *b*) who use a spatially anisotropic refinement strategy with no temporal refinement and by Edwards (1996) who uses a cell-by-cell refinement strategy.

The focus of this paper is on developing a structured-grid adaptive mesh refinement (AMR) algorithm for porous media flow. Our approach is similar to the approach introduced by Hornung and Trangenstein (1997) and to the approach discussed by Propp (1998). In these approaches, the solution is represented on a hierarchical sequence of nested grids with successively finer spacing in both time and space. Increasingly finer grids are recursively embedded in coarse grids until the solution is sufficiently resolved. An error estimation procedure based on user-specified criteria evaluates where additional refinement is needed and grid generation procedures dynamically create or remove rectangular fine grid patches as resolution requirements change. The method presented here uses subcycling in time so that all levels are advanced at the same CFL number, thus reducing the numerical dissipation of the explicit upwind advection scheme used to advance the solution. The major difference between the approach adopted here and that of Hornung and Trangenstein (1997) is that the current method does not require a global, multilevel pressure solve at each fine-grid time step. Instead, when advancing a given level, we solve the pressure on that level only with boundary conditions obtained from the coarser levels. This approach avoids the computational expense of the global solve but introduces additional complexity into the synchronization step of the algorithm in which inconsistencies between different levels are corrected. The synchronization approach used here is based on the algorithm developed by Almgren *et al.* (1998) for incompressible flows. The methodology is implemented in parallel using the BoxLib framework discussed in Rendleman *et al.* (2000) and Crutchfield (1991). A similar approach was used by Propp (1998); however, his algorithm used a different approach to synchronization and was limited to two dimensions.

Before describing the adaptive algorithm we will briefly review the total velocity splitting approach and discuss our basic fractional step scheme for a single grid. In the third section we describe, in detail, the recursive time-stepping procedure for the adaptive algorithm and other aspects of the adaptive algorithm. The fourth section shows convergence results and presents computational examples illustrating both the performance and parallel scaling of the method.

2. Total velocity-splitting algorithm

Here we consider multiphase, multicomponent incompressible flow in heterogeneous porous media. The multicomponent mixture is composed of N components (or lumped pseudo-components). We define $\mathbf{n} \equiv (n_1, \ldots, n_N)$ as the vector of component densities per unit pore volume, and let $\mathbf{n}_\alpha$ represent the portion of $\mathbf{n}$ in phase α, where Greek subscripts refer to mobile phases. Thus, $\sum_\alpha \mathbf{n}_\alpha \equiv \mathbf{n}$ is the total component density in the combined fluid system. In a general setting, the separation of components into phases requires a computation of thermodynamic equilibrium for the mixture. For the problems considered here, the phases are incompressible, we assume no volume change on mixing and each component only appears in a single phase. The basic void fraction of the medium is referred to as the porosity, ϕ,

and the fraction of that void occupied by a particular phase is referred to as the phase saturation, u_α, which, for the models considered here, can be determined from the composition of the phase, $\mathbf{n}_\alpha$, and the pure component densities.

The equations that describe the flow represent mass conservation, energy conservation, and Darcy's law. Darcy's law expresses the volumetric flow rate, v_α, of each phase in terms of the phase pressure, p_α, namely,

$$v_\alpha = -\lambda_\alpha(\nabla p_\alpha - \rho_\alpha \vec{g}) \ , \tag{2.1}$$

where $\lambda_\alpha \equiv K k_{r,\alpha}/\mu_\alpha$ is the phase mobility. Here K is the permeability distribution of the medium; $k_{r,\alpha}$ is the relative permeability, which is a function of u_α that expresses modification of the flow rate due to multiphase effects; μ_α and ρ_α are the phase viscosity and phase density, respectively, which depend on the phase composition; and $\vec{g}$ is the gravitational acceleration. The pressure in each phase is related to a reference pressure, p, by a capillary pressure, $p_{c,\alpha} = p_\alpha - p$, which is a function of saturation.

For this system, conservation of mass for each component is given by

$$\phi \frac{\partial \mathbf{n}}{\partial t} + \nabla \cdot \sum_\alpha \frac{\mathbf{n}_\alpha}{u_\alpha} v_\alpha = \nabla \cdot \mathcal{D} + \mathcal{R}, \tag{2.2}$$

where $\mathcal{D}$ represents diffusive terms that include multiphase molecular diffusion and dispersion, and $\mathcal{R}$ is the reaction source term. The diffusive term associated with capillary pressure is implicitly included in the definition of the phase velocities. The diffusion terms can be quite complex, depending on the particular problem; see the work of Xu *et al.* (2004) for a detailed discussion of these terms. For purposes of exposition, we will set $\mathcal{D} = 0$ and $\mathcal{R} = 0$ for the remainder of the discussion. However, we note that within this framework additional diffusion terms can be added analogously to the capillary pressure term. If reactions are included in the system they can be included using the operator-split formalism discussed in Day and Bell (2000).

The component conservation equations specify the change in total mass of each component resulting from the transport and diffusion of that component distributed across the phases. The phase behavior of the system specifies how the components are apportioned into phases and the volume occupied by those phases. In this paper, we consider only simple incompressible systems without mass transfer between phases. We can then define a total velocity

$$v_T \equiv \sum_\alpha v_\alpha = -\sum_\alpha \lambda_\alpha(\nabla p_\alpha - \rho_\alpha \vec{g}) = -\sum_\alpha \lambda_\alpha(\nabla(p + p_{c,\alpha} - \rho_\alpha \vec{g}), \tag{2.3}$$

since $p_\alpha = p + p_{c,\alpha}$. The total velocity is divergence-free so that

$$\nabla \cdot v_T = -\nabla \cdot \sum_\alpha \lambda_\alpha(\nabla(p + p_{c,\alpha}) - \rho_\alpha \vec{g}) = 0 \quad . \tag{2.4}$$

This leads to a second-order elliptic equation for pressure:

$$\nabla \cdot \sum_\alpha \lambda_\alpha \nabla p = -\nabla \cdot \left(\sum_\alpha \lambda_\alpha \nabla p_{c,\alpha} - \rho_\alpha \vec{g} \right) \quad . \tag{2.5}$$

We can rewrite (2.3) to express ∇p in terms of v_T, then using (2.1) we can express v_α in terms of the total velocity,

$$v_\alpha = \frac{\lambda_\alpha}{\lambda_T} v_T + \lambda_\alpha(\rho_\alpha - \sum_\beta \frac{\lambda_\beta \rho_\beta}{\lambda_T})\vec{g} - \lambda_\alpha(\nabla p_{c,\alpha} - \sum_\beta \frac{\lambda_\beta \nabla p_{c,\beta}}{\lambda_T}) \ ,$$

where $\lambda_T = \sum_\alpha \lambda_\alpha$ is the total mobility. Writing the conservation equations in terms of the total velocity yields the fractional flow form of the component conservation equations, given by

$$\phi \frac{\partial \mathbf{n}}{\partial t} + \nabla \cdot F(\mathbf{n}, v_T) = \nabla \cdot (H \nabla P_c), \tag{2.6}$$

where

$$F(\mathbf{n}, v_T) = \sum_\alpha \frac{\mathbf{n}_\alpha}{u_\alpha} \left[\frac{\lambda_\alpha}{\lambda_T} v_T + \lambda_\alpha(\rho_\alpha - \sum_\beta \frac{\lambda_\beta \rho_\beta}{\lambda_T})\vec{g} \right] ,$$

and

$$H \nabla P_c = \sum_\alpha \frac{\mathbf{n}_\alpha}{u_\alpha} \lambda_\alpha(\nabla p_{c,\alpha} - \sum_\beta \frac{\lambda_\beta \nabla p_{c,\beta}}{\lambda_T}).$$

In the absence of diffusive terms the total velocity defines a splitting that decomposes the dynamics into an elliptic pressure equation and a system of hyperbolic conservation laws. We note that we are assuming here that the system $\phi \frac{\partial \mathbf{n}}{\partial t} + \nabla \cdot F(\mathbf{n}, v_T) = 0$ is hyperbolic with v_T viewed as a function of space and time. This is a condition on the model specification; some models for three-phase flow use relative permeability models that lead to non-hyperbolic behavior and the resulting problems are ill-posed in the zero-viscosity limit (Bell *et al.*, 1986).

Single grid algorithm

In this subsection, we discuss the single grid algorithm, giving an overview of the time stepping procedure. The discretization uses a volume of fluid approach in which $\mathbf{n}^n_{ijk}$ denotes the average value of $\mathbf{n}$ over cell (i, j, k) at time t^n; $\mathbf{n}$ and p are defined on cell centers while F and v_T are defined on cell edges. The temporal discretization fits into the basic framework of IMPES-type methods in which the pressure equation is solved implicitly and the saturation (component) equations are solved explicitly. Here the basic algorithm is modified so that the overall splitting approach is second-order accurate in time and we treat the diffusion terms semi-implicitly so that diffusive terms, $H \nabla P_c$, do not limit the time step.

The outline of the algorithm is as follows:

- **Step 1:** Solve the pressure equation, (2.5), rewritten in the form

$$D \sum_\alpha \lambda_\alpha (Gp) = D \sum_\alpha \lambda_\alpha \rho_\alpha \vec{g} - D \sum_\alpha \lambda_\alpha (G p_{c,\alpha}) \ ,$$

  for p with properties evaluated using $\mathbf{n}^n$. We then use equation (2.3) to define a total velocity v^n_T. Here D and G are second-order accurate discretizations of the divergence and gradient operators, respectively. The divergence operator

returns a cell-centered divergence from face-centered values; the gradient operator differences cell-centered values to return normal gradients on faces. For example, in two dimensions the discretization of $D\sum_\alpha \lambda_\alpha(Gp)$ at cell (i,j) would be

$$\sum_\alpha \left(\frac{(\lambda_{\alpha_{i+\frac{1}{2},j}}(p_{i+1,j}-p_{i,j}) - \lambda_{\alpha_{i-\frac{1}{2},j}}(p_{i,j}-p_{i-1,j}))}{\Delta x} + \frac{(\lambda_{\alpha_{i,j+\frac{1}{2}}}(p_{i,j+1}-p_{i,j}) - \lambda_{\alpha_{i,j-\frac{1}{2}}}(p_{i,j}-p_{i,j-1}))}{\Delta y} \right)$$

where Δx and Δy are the mesh spacings in the x and y directions, respectively.

- **Step 2:** Use (2.6) to advance the solution from time t^n to time t^{n+1} using v_T^n. We use an unsplit second-order Godunov scheme to compute the hyperbolic fluxes using the methodology described in Bell *et al.* 1989. (See Almgren *et al.* 1998 for additional details of the multidimensional aspects of the advection scheme.) The Godunov discretization is coupled to a Crank-Nicolson discretization of the diffusive terms, so that

$$\phi\frac{\mathbf{n}^{n+1,*}-\mathbf{n}^n}{\Delta t} + DF^{n+1/2,*} = 1/2\left(D(H^nGP_c^n) + D(H^{n+1,*}GP_c^{n+1,*})\right) \quad (2.7)$$

  with a suitable linearization of the coefficients of the diffusion term; here we use $H^{n+1,*}$ and $P_c^{n+1,*}$ to denote H and P_c, respectively, which are functions of $\mathbf{n}$, evaluated at $\mathbf{n}^{n+1,*}$. In this step, $F^{n+1/2,*}$ denotes time-centered fluxes computed using the Godunov procedure but with the total velocity evaluated at t^n.

- **Step 3:** Solve the pressure equation (2.5), with properties evaluated using $\mathbf{n}^{n+1,*}$, to compute a new total velocity v_T^{n+1} from (2.3). We then define $v_T^{n+1/2} = 1/2(v_T^n + v_T^{n+1})$ so that we can time center the dependence of the flux on v_T.

- **Step 4:** Use (2.6) to re-advance the solution from time t^n to time t^{n+1}, this time using $v_T^{n+1/2}$ to obtain values of $\mathbf{n}^{n+1}$,

$$\phi\frac{\mathbf{n}^{n+1}-\mathbf{n}^n}{\Delta t} + DF^{n+1/2} = 1/2\left(D(H^nGP_c^n) + D(H^{n+1}GP_c^{n+1})\right) \quad (2.8)$$

  again with a suitable linearization of the coefficients of the diffusion term. In this step, $F^{n+1/2}$ denotes time-centered fluxes computed using the Godunov procedure but with the total velocity evaluated at $t^{n+1/2}$.

We note here that Step 1 is not always necessary; it can be acceptable to replace v_T^n with the final v_T computed in Step 3 of the previous time step without significant loss of accuracy, thus allowing us to avoid the computational cost associated with Step 1. In this case, v_T^n is computed with $\mathbf{n}^{n,*}$ instead of $\mathbf{n}^n$. We shall demonstrate in Section 4 that for our examples this has no noticeable effects on our solution. For clarity, we shall denote a time step that uses Step 1 as a *full solve* and one without as a *partial solve.*

3. Adaptive Mesh Refinement

In this section we present the extension of the algorithm described above to an adaptive hierarchy of nested rectangular grids. First we describe the creation of the grid hierarchy and the regridding procedure used to adjust the hierarchy during the computation. Next we discuss the adaptive time step algorithm with subcycling in time, focusing on the synchronization between different levels of refinement.

(*a*) *Creating and Managing the Grid Hierarchy*

The grid hierarchy is composed of different levels of refinement ranging from coarsest ($\ell = 0$) to finest ($\ell = \ell_{max}$). Each level is represented as the union of rectangular grid patches of a given resolution. In this implementation, the refinement ratio is always even, with the same factor of refinement in each coordinate direction, i.e. $\Delta x^{\ell+1} = \Delta y^{\ell+1} = \Delta z^{\ell+1} = \frac{1}{r}\Delta x^{\ell}$, where r is the refinement ratio. (We note here that neither isotropic refinement nor uniform base grids are requirements of the fundamental algorithm.) In the actual implementation, the refinement ratio, either 2 or 4, can be a function of level; however, in the exposition we will assume that r is constant. The grids are properly nested, in the sense that the union of grids at level $\ell + 1$ is contained in the union of grids at level ℓ for $0 \leq \ell < \ell_{max}$. Furthermore, the containment is strict in the sense that, except at physical boundaries, the level ℓ grids are large enough to guarantee that there is a border at least one level ℓ cell wide surrounding each level $\ell+1$ grid. (Grids at all levels are allowed to extend to the physical boundaries so the proper nesting is not strict there.)

The initial creation of the grid hierarchy and the subsequent regridding operations in which the grids are dynamically changed to reflect changing flow conditions use the same procedures as were used by Bell *et al.* (1994) for hyperbolic conservation laws. The construction of the grid hierarchy is based on error estimation criteria specified by the user to indicate where additional resolution is required. The error criteria are currently based on tracking component density gradients for one of the components; however, more sophisticated criteria based on estimating the error can be used (see, e.g., Berger and Colella (1989)). Given grids at level ℓ, we use the error estimation procedure to tag cells where the criteria for further refinement are met. We then tag a buffer region n_{buf} cells wide around the originally tagged cells so that the features of interest are safely contained within the newly created fine level. The tagged cells are grouped into rectangular patches using the clustering algorithm given in Berger and Rigoutsos (1991). These rectangular patches are refined to form the grids at the next level. The process is repeated until either the error tolerance criteria are satisfied or a specified maximum level is reached. The proper nesting requirement is imposed at this stage.

At $t = 0$, the initial data is used to create grids at level 0 through ℓ_{max}. (Grids have a user-specified maximum size, therefore more than one grid may be needed to cover the physical domain.) As the solution advances in time, the regridding algorithm is called every k_ℓ (also user-specified) level ℓ steps to redefine grids at levels $\ell + 1$ to ℓ_{max}. The parameter k_ℓ should reflect the constraint that we do not want the tagged feature to move off level $\ell + 1$ during k_ℓ level $\ell + 1$ time steps, so typically $k_\ell \leq n_{buf}$. Level 0 grids remain unchanged throughout the calculation. Grids at level $\ell + 1$ are only modified at the end of level ℓ time steps, but because

we subcycle in time, i.e. $\Delta t^{\ell+1} = \frac{1}{r}\Delta t^{\ell}$, level $\ell + 2$ grids can be created and/or modified in the middle of a level ℓ time step if $k_{\ell+1} < r$.

When new grids are created at level $\ell+1$, the data on these new grids are copied from the previous grids at level $\ell + 1$ if possible, or interpolated in space from the underlying level ℓ grids otherwise. After regridding we always recalculate v_T from the new data, thus the first fine time step after regridding is always a *full solve* rather than a *partial solve*. We note here that while there is a user-specified limit to the number of levels allowed, at any given time in the calculation there may not be that many levels in the hierarchy, i.e. ℓ_{max} can change dynamically as the calculation proceeds, as long as it does not exceed the user-specified limit.

(*b*) *Overview of Time-Stepping Procedure*

The adaptive time stepping approach uses temporal subcycling so that each level is advanced independently at its own time step ($\Delta t^{\ell+1} = \frac{1}{r}\Delta t^{\ell}$), requiring no interlevel communication other than the supplying of Dirichlet data from a coarse level to be used as boundary conditions at the next finer level. When coarse and fine data reach the same point in time, the data at the different levels are synchronized. The synchronization approaches are based on the same set of algorithmic ideas as those developed in Almgren *et al.* (1998)

The adaptive time-step algorithm can most easily be thought of as a recursive procedure, in which to advance level ℓ, $0 \leq \ell \leq \ell_{max}$ the following steps are taken:

- Advance level ℓ in time as if it is the only level. Use boundary conditions for component densities and pressure from level $\ell - 1$ if level $\ell > 0$, and from the physical domain boundaries.
- If $\ell < \ell_{max}$
  - Advance level $(\ell+1)$ r times with time step $\Delta t^{\ell+1} = \frac{1}{r}\Delta t^{\ell}$. Use boundary conditions for component densities and pressure from level ℓ and from the physical domain boundaries.
  - Synchronize the data between levels ℓ and $\ell + 1$, and interpolate corrections to higher levels if $\ell + 1 < \ell_{max}$.

Before describing the steps of the synchronization in detail, we first discuss, in general terms, how to synchronize the data at different levels so that the solution as computed on each level sequentially can most closely approximate the solution which would be found using composite solves. During the advance of each level, for each operator we supply Dirichlet boundary data for the fine grids from the next coarser grid. This implies that the values at both levels are consistent, but the computed fluxes at the coarse/fine interfaces are not. The synchronization solves correction equations that account for discrepancies in fluxes between levels. The correction equations reflect the type of operator being corrected. For hyperbolic equations the correction of flux discrepancies is a simple explicit flux correction as discussed in Berger and Colella (1989). For a self-adjoint elliptic operator, the discrepancy in the fluxes represents a discontinuity in normal derivative at the coarse/fine boundary and the correction equation takes the form of a discrete layer potential problem (Almgren *et al.*, 1998).

After the level $\ell+1$ data have been advanced to the same point in time as the level ℓ data, there are three sources of discrepancy in the composite solution that need to be corrected in the synchronization step.

(D.1) The data at level ℓ that underlie the level $\ell+1$ data are not synchronized with the level $\ell+1$ data.

(D.2) The composite total velocity computed from the pressure equation, defined as the time-averaged (over a level ℓ time step) level $\ell+1$ advection velocity on all level $\ell+1$ faces including the $\ell|(\ell+1)$ interface, and the level ℓ advection velocity on all other level ℓ faces, does not satisfy the composite divergence constraint at the $\ell|(\ell+1)$ interface.

(D.3) The advective and diffusive fluxes from the level ℓ faces and the level $\ell+1$ faces do not agree at the $\ell|(\ell+1)$ interface, resulting in a loss of conservation.

The aim of the synchronization steps is to correct the effects of each mismatch. Discrepancy (D.1) is easily corrected by averaging $\mathbf{n}$ at level $\ell+1$ onto level ℓ. We denote this correction by (S.1). The third discrepancy, (D.3), is also easily corrected in the case without diffusive terms; we simply correct the solution in the coarse cells immediately adjacent to the fine level by the divergence of an edge-based correction field that is non-zero only on the coarse/fine interface where it contains the difference between the coarse flux and time- and space-averaged fine flux. In combination with (S.1) this ensures overall conservation.

The second discrepancy, (D.2), is discretely manifest as a non-zero difference between the coarse grid total velocity and the effective time- and space-averaged fine grid total velocity at the coarse/fine interface. This difference results from not having satisfied the elliptic matching conditions at the coarse/fine interface during the pressure solve. An elliptic solve is necessary to correct for the discrepancy. We perform a level ℓ "pressure sync solve," (S.2) with the right-hand-side defined as the divergence of the mismatch between the level ℓ and the time-averaged level $\ell+1$ total velocity. From this solve we define a total velocity correction field that is used to modify the explicit dependence of the hyperbolic flux terms on the total velocity. In the case of zero diffusive terms these "re-advection corrections" are explicitly added to the new-time solution in all cells at level ℓ and are interpolated to the new-time solution at all higher levels.

In the case of non-zero diffusive terms, the modification of the solution by the refluxing (S.2) and re-advection (S.3) corrections requires solving additional elliptic equations. These will be described in more detail in the next subsection.

(*c*) *Details of Time-Stepping Procedure*

Assume now that we are advancing level ℓ, $0 \le \ell \le \ell_{max}$, one level ℓ time step. We now add the level index, ℓ, to the superscript of each quantity, defining, for example, $\mathbf{n}^{n,\ell}$ as the component density on level ℓ at time t^n. We define Δt^ℓ as the time step of level ℓ.

(i) *Advancing a single level*

To advance the data on level ℓ one level ℓ time step, we follow the time-stepping procedure as described for the single grid algorithm in the previous section. For the

level ℓ advancement we advance all the grids at that level simultaneously. Explicit advection work is decoupled except for the exchange of boundary conditions. When a coarse/fine boundary does not coincide with a physical domain boundary for the level ℓ advection step, level $\ell - 1$ data are interpolated linearly in time to specify Dirichlet boundary conditions at the coarse-fine boundary. The procedures used for these interpolations are the same as those discussed in Almgren *et al.* (1998).

(ii) *Computing the Coarse-Fine Discrepancy*

Over the course of a level ℓ time step, we must accumulate several quantities at the $\ell|(\ell+1)$ interface in order to correctly capture the flux discrepancies at the end of the level ℓ time step. We refer to the face-based data structures that contain these quantities as registers. The total velocity and flux registers accumulate the discrepancies between the level ℓ and level $\ell+1$ face-based total velocity v_T and fluxes F, respectively.

These registers are defined only on the $\ell|(\ell+1)$ interface and are indexed by level ℓ indices. Note that in d dimensions, one level ℓ face contains r^{d-1} level $\ell+1$ faces; the sums over faces below should be interpreted as summing over all level $\ell+1$ faces which are contained in the level ℓ face. The sums over k should be understood as summing over the r level $\ell+1$ time steps contained within a single level ℓ time step.

At the end of the level ℓ time step, the velocity register (δv_T^ℓ) holds the difference between the total velocity at level ℓ and the time average over one level ℓ time step of the space average over the area of the level ℓ face of the total velocity at level $\ell+1$:

$$\delta v_T^\ell = -v_T^{n+1/2,\ell} + \frac{1}{r^d}\sum_{k=1}^{r}\sum_{faces} v_T^{k+1/2,\ell+1}.$$

We note that the velocity included in the total velocity register is the time-centered velocity, $v_T^{n+1/2}$, that is defined in Step 3 as the time average of v_T^n and v_T^{n+1} because this is the velocity field used in advection.

The flux registers, $\delta F^{hyp,\ell}$ and $\delta F^{diff,\ell}$, contain the differences between the hyperbolic and diffusive fluxes calculated at level ℓ and the time average over the level ℓ time step of the space average over the area of the level ℓ face of the advective fluxes at level $\ell+1$:

$$\delta\mathcal{F}^{hyp,\ell} = -\mathcal{F}^{hyp,\ell} + \frac{1}{r^d}\sum_{k=1}^{r}\sum_{faces}\mathcal{F}^{hyp,k,\ell+1},$$

$$\delta\mathcal{F}^{diff,\ell} = -\mathcal{F}^{diff,\ell} + \frac{1}{r^d}\sum_{k=1}^{r}\sum_{faces}\mathcal{F}^{diff,k,\ell+1},$$

where

$$\begin{aligned}\mathcal{F}^{hyp,\ell} &= F^{n+1/2,\ell},\\ \mathcal{F}^{diff,\ell} &= 1/2\left(H^{n,\ell}GP_c^{n,\ell} + H^{n+1,\ell}GP_c^{n+1,\ell}\right),\end{aligned}$$

as computed in Step 4 of the algorithm.

We note here that the signs of the quantities added to the flux registers actually depend on the orientation of the normal facing away from the fine grid. We follow the convention below that the signs are given for the faces at which the fine grid is in the direction of the lower coordinate indices.

(iii) *Synchronization of data*

The first synchronization step, (S.1), has already been described in subsection (b). Here we give the details of the other two steps of the synchronization.

The discrepancy in the total velocity, (D.2), is captured in δv_T^ℓ; the divergence of δv_T^ℓ defines the right-hand-side for the level ℓ elliptic sync solve (S.2). We solve

$$-D\left(\lambda_T G(\delta e^\ell)\right) = \widetilde{D}(\delta v_T^\ell)$$

on all grids at level ℓ for the correction δe^ℓ. Recall δv_T^ℓ is defined only at the coarse/fine interface; here $\widetilde{D}$ is defined to be the discrete divergence operator evaluated only on the level ℓ cells adjacent to the interface but not underlying any level $\ell+1$ grids, and including contributions only from the coarse/fine interface. Boundary conditions on physical no-flow boundaries are homogeneous Neumann ($\frac{\partial(\delta e)^\ell}{\partial n} = 0$); on outflow $\delta e^\ell = 0$. If $\ell > 0$, the boundary conditions for δe^ℓ are given as homogeneous Dirichlet conditions on the level $\ell-1$ cells outside the level ℓ grids. We then define the correction velocity field from δe^ℓ :

$$v_T^{corr,\ell} = -\lambda_T G(\delta e^\ell).$$

We now use $(v_T^{n+1/2,\ell} + v_T^{corr,\ell})$ to define new fluxes on all level ℓ faces, and define $\mathcal{F}_{v_T}^{corr,\ell}$ as the difference between the original fluxes and these newly computed fluxes. In the case of non-zero diffusive terms, we must diffuse the re-advection and refluxing corrections before adding them to the new-time solution, so we do not add them directly to the solution. Rather, the divergence of the re-advection flux corrections is added to the advective and diffusive flux mismatches to define the cell-centered right-hand-sides for the refluxing solves (S.3):

$$RHS_{sync}^\ell = -D\mathcal{F}_{v_T}^{corr,\ell} - \widetilde{D}(\delta\mathcal{F}^{hyp,\ell} + \delta\mathcal{F}^{diff,\ell}). \tag{3.1}$$

Then, we solve for the correction to the solution, $\mathbf{n}_{sync}^\ell$

$$\phi\mathbf{n}_{sync}^\ell - \frac{\Delta t}{2} D(H^s G P_c^s) = RHS_{sync}^\ell \ . \tag{3.2}$$

where $H^s = H(\mathbf{n}^{n+1,\ell} + \mathbf{n}_{sync}^\ell)$ and $P_c^s = P_c(\mathbf{n}^{n+1,\ell} + \mathbf{n}_{sync}^\ell)$.

If $\ell > 0$, we must now modify the level $\ell-1$ velocity registers and flux registers to account for the corrections to the solution due to the re-advection corrections as well as the diffused corrections. This is analogous to the accumulation of advective

and diffusive fluxes during the advance of a single level. To do this, we set

$$\begin{aligned}
\delta v_T^{\ell-1} &:= \delta v_T^{\ell-1} + \frac{1}{r^d} \sum_{faces} v_T^{\ell,corr}, \\
\delta \mathcal{F}^{hyp,\ell-1} &:= \delta F^{hyp,\ell-1} + \frac{1}{r^d} \sum_{faces} \mathcal{F}^{corr,\ell}, \\
\delta \mathcal{F}^{diff,\ell-1} &:= \delta \mathcal{F}^{diff,\ell-1} + \frac{1}{r^d} \sum_{faces} \left\{ \frac{1}{2} H^s G P_c^s \right\}.
\end{aligned}$$

These will enter into the correction performed between levels $\ell-1$ and ℓ at the end of the level $\ell-1$ time step if $\ell > 1$.

The corrected solution at level ℓ is then given by

$$\mathbf{n}^{n+1,\ell} := \mathbf{n}^{n+1,\ell} + \mathbf{n}^{\ell}_{sync}.$$

If $\ell < \ell_{max}$, we interpolate the correction onto the fine grids at all finer levels, q, $\ell < q \leq \ell_{max}$ using conservative interpolation:

$$\mathbf{n}^{n+1,q} := \mathbf{n}^{n+1,q} + \mathrm{Interp}_{cons}(\mathbf{n}^{\ell}_{sync}).$$

This completes the synchronization steps for scalar quantities.

4. Numerical Results

In this section, we examine the numerical properties of the method described in the previous section. In the first subsection we examine the convergence behavior of the method. While the basic discretization presented here is similar to a traditional IMPES fractional step discretization, we have modified the method such that the total velocity is centered in time. This makes the method formally second-order accurate, in contrast to the standard IMPES approach. In our first example, we use a stable one-dimensional problem based on a two-component single-phase system to demonstrate the method's second-order rate of convergence and to illustrate the loss of accuracy associated with the standard IMPES-type fractional step scheme. We then demonstrate the second-order convergence rate of the algorithm with a two-dimensional problem and show that the *partial solve* algorithm maintains the overall accuracy of the method relative to the *full solve* version. We also examine the effects of centering the total velocity and compare the current approach with the standard IMPES discretization when the flow is unstable. In the second subsection, we compare solutions obtained on a uniform grid with those computed with AMR. For this purpose, we use a more complicated example involving a three-component two-phase system. In the final two subsections, we simulate a large three-dimensional problem using the parallelized AMR algorithm, and present the scaling behavior. We note that for the numerical results presented here, we neglect capillary pressure.

(*a*) *Convergence*

For our first study we consider a single-phase two-component system where components 1 and 2 are completely miscible, and form a mixture with viscosity

Table 1. L^1 *errors and convergences rate of the method for a 1D two-component single-phase system.*

h, m	IMPES	rate	NEW	rate
2/50	1.06e-2	–	9.92e-3	–
2/100	2.86e-3	1.89	2.44e-3	2.02
2/200	1.25e-3	1.19	5.97e-4	2.03
2/400	6.07e-4	1.04	1.50e-4	1.99
2/800	3.00e-4	1.01	3.58e-5	2.07

given by

$$\mu = \frac{1}{(1 - c + M^{1/4}c)^4} \text{ cP} \ , \tag{4.1}$$

where $c = n_1/(n_1 + n_2)$ and $M = 10$. In our algorithm as presented, both the pressure discretization and the component conservation equations are discretized to second-order accuracy. In the standard IMPES formulation, the total velocity is not centered in time, which introduces a small first-order temporal error. In most realistic problems, this error is typically dominated by spatial truncation errors, so we examine the problem in the context of one dimension where we can use fine spatial resolution. We take as initial data $c(x) = 0.5(1+\tanh((0.3-x)/0.05)$ on the interval $[0, 2]$ and impose a constant pressure drop. The time step, which is fixed for each resolution, is set so that the CFL is initially approximately 0.35. In addition, we turn off the flux limiting in the advection scheme.

The L^1 norm of the errors for the standard IMPES discretization and the *full solve* version of the new algorithm are presented in Table 1. The new algorithm shows clear second-order convergence whereas the standard IMPES algorithm is only first-order accurate asymptotically. At the coarsest resolution, the two approaches have nearly the same error and the IMPES discretization shows nearly second-order convergence between the two coarsest discretizations. At these coarser resolutions the temporal discretization error is dominated by the spatial truncation error, which is second-order accurate in both cases.

Next, we examine the convergence properties of the single grid algorithm in two dimensions. For this problem we consider a domain $[0, l_x] \times [0, l_y]$ where $l_x = 16$m and $l_y = 4$m. At time $t = 0$, the domain contains only component 2. For time $t > 0$, component 1 flows into the domain from the left edge of the domain with a composition given by $n_1(0, y; t) = \tau\rho_1$ and $n_2(0, y; t) = (1 - \tau)\rho_2$ where $\tau(t) = 0.5(1 + \tanh(t/10^6)))$. The densities of component 1 and 2 are 1000 kg/m^3 and 800 kg/m^3, respectively. We impose no-flow conditions on the top and bottom edges and a pressure difference of 2 atm between the inlet and the outlet. We define $M = 2$. The porosity is uniformly 1 while the permeability function $\kappa(x, y)$ is given by

$$\kappa(x, y) = 100(1 + \sin(\pi y/l_y)^4) \text{ mD}.$$

We note that $\tau(t)$ and $\kappa(x, y)$ are chosen so that the solution will be smooth. Gravity points in the negative y direction. The calculation is run for 1.2×10^7 s with $\Delta x = \Delta y = 1/2^n$ m for $n = 3, 4, 5, 6$. Uniform grids and a fixed time step are used for the purpose of evaluating the convergence of the solution.

Table 2 shows the discrete L^∞, L^1 and L^2 norms of the difference between the solution n_1 obtained on each grid and that obtained on the next finer grid, and

Table 2. L^∞, L^1 *and* L^2 *errors and converge rates for* n_1 *in a 2-D two-component single-phase calculation with the* partial solve *algorithm.*

n	3	rate	4	rate	5	rate	6
L^∞	7.980	1.38	3.0612	1.57	1.29	1.11	0.5960
L^1	0.752	1.91	2.005e-1	1.99	5.047e-2	1.95	1.306e-2
L^2	1.438	1.83	4.036e-1	1.88	1.097e-1	1.81	3.137e-2

Table 3. L^∞, L^1 *and* L^2 *errors and converge rates for* n_1 *in a 2-D two-component single-phase calculation, with the* full solve *approach.*

n	3	rate	4	rate	5	rate	6
L^∞	7.978	1.38	3.061	1.29	1.250	1.08	0.5913
L^1	0.752	1.91	2.006e-1	1.99	5.050e-2	1.98	1.282e-2
L^2	1.439	1.83	4.039e-1	1.88	1.098e-1	1.81	3.128e-2

the resulting convergence rates. The rate between the two columns of error norms is defined as $\log_2(\varepsilon_l/\varepsilon_r)$ where ε_l and ε_r are the errors shown in the left and right columns. Table 2 clearly demonstrates the second-order convergence property of the algorithm when measured in the L^1 and L^2 norms. The convergence rate of the L^∞ error is less than 2; this can be attributed to limiters used in the Godunov scheme. Table 3 shows the same data but for calculations done using the *full solve* approach. We see by comparing the two tables that the errors obtained from the different approaches are very similar. We conclude that the omission of Step 1 does not adversely affect the convergence rate of our method.

The next set of calculations compares the solution obtained using our method to that obtained using a first-order method when the solution is no longer smooth. The goal is to show that even though the flow is no longer in the asymptotically second-order regime of the algorithm, the accuracy properties of the formally second-order method still have important consequences for the behavior of the flow. We construct the first-order method by eliminating Steps 3 and 4 from the method presented in Section 2, and setting $\mathbf{n}^{n+1} = \mathbf{n}^{n+1,*}$.

However, the error in the solution cannot be directly measured by taking the norm of the difference between two solutions. We instead compare the mixing length given by the length of the fingered zone. Following the work of Manickam and Homsy (1995), we define the mixing length, L_δ, as

$$L_\delta = x|_{\bar{c}=\delta} - x|_{\bar{c}=1-\delta} \tag{4.2}$$

where

$$\bar{c}(x) = \frac{\displaystyle\int_0^{l_y} n_1 dy}{\displaystyle\int_0^{l_y} n_1 dy + \int_0^{l_y} n_2 dy}$$

and $\delta = 1\times10^{-5}$. We first look at a stable fingering profile that allows us to establish a hypothesis on how mixing length is related to accuracy. We then examine an example where the flow is highly unstable.

For the first example, we again look at the two-component single-phase system described in the previous section. We consider here a uniform $\kappa(x, y) = 200$ mD and a pressure difference of 0.5 atm between the inlet and the outlet. We see in

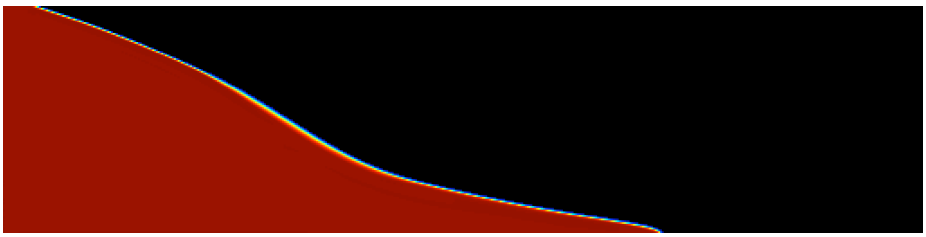

Figure 1. Solution of n_1 at 8×10^6 s.

Table 4. *Comparison of the mixing length L_δ at time 8×10^6 s; ϵ_r is the relative error when compared to solution obtained for grid size 1024×256 with the* full solve *second-order method.*

grid size	512×128		1024×256		2048×512	
	L_δ, m	ϵ_r	L_δ, m	ϵ_r	L_δ, m	ϵ_r
1st-order	11.0313	7.23e-2	11.5000	3.29e-2	11.8828	6.57e-4
2nd-order, partial	11.0625	6.96e-2	11.5156	3.16e-2	11.8906	0
2nd-order, full	11.0625	6.96e-2	11.5156	3.16e-2	11.8906	–

Figure 1 that a stable finger extends from the inlet to the outlet at the bottom of the simulation domain. We then measure the mixing length for three different grid sizes and three different approaches. The grids we consider are 512×128, 1024×256 and 2048×512. The three approaches are first-order, second-order with the *partial solve* algorithm, and second-order with the *full solve* approach. The solution obtained from the 2048×512 grid and the *full solve* approach is used as the reference solution. The simulation is run to a final time of 8×10^6 s.

To amplify the difference between the first-order method and the second-order methods, we change the permeability distribution function $\kappa(x, y)$ to

$$\kappa(x,y) = \begin{cases} 150 \text{ mD}, & y/l_y < .25; \\ 300 \text{ mD}, & 0.25 \le y/l_y < .5; \\ 100 \text{ mD}, & 0.5 \le y/l_y < .75; \\ 200 \text{ mD}, & y/l_y \ge .75. \end{cases} \tag{4.3}$$

In addition, M is increased to 10, dramatically increasing the inherent instability of the problem. Figure 2 shows that the viscous fingers resulting from the first-order method and the second-order methods have some notable differences. Slight variations in the solution can develop into distinct fingering profiles due to instabilities in the solution. To further illustrate this point, we examine a two-component single-phase system in a domain where $\kappa = 100\ (1 + 10^{-8}\epsilon)$ mD and ϵ is a random number between 0 and 1. Figure 3 shows that two different realizations of ϵ lead to two different fingering profiles. This suggests that any minor perturbation can lead to variation in the details of the fingering. In spite of this sensitivity, we see that the omission of Step 1 does not have significant effects on the solution. In Table 5, we list the mixing lengths for grid sizes 512×128 and 1024×256 computed with the three approaches. The minor differences in mixing length for the two second-order methods further demonstrate that the additional work required

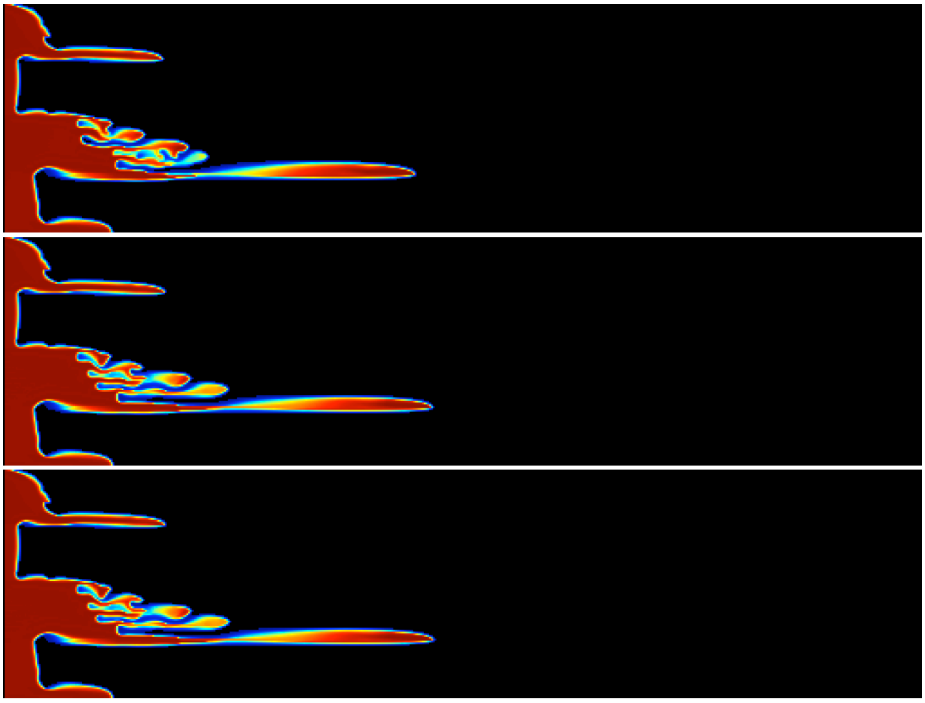

Figure 2. Concentration of component 1 after time 5×10^6 s based on first order method (top), second order method (middle), and second order method with Step 1 (bottom).

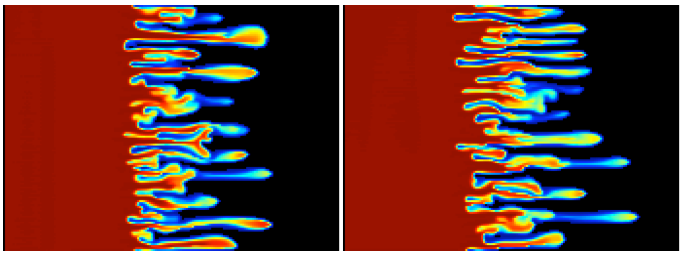

Figure 3. Fingering details of two different perturbation profiles of κ.

Table 5. *Comparison of the mixing length L_δ at time 5×10^6 s for an unstable two-component single-phase system.*

grid size	L_δ, m	
	512×128	1024×256
1st-order	7.219	7.906
2nd-order, partial	7.500	8.219
2nd-order, full	7.531	8.250

for the *full solve* algorithm is unnecessary. Results of the subsequent sections are based on the second-order method with the *partial solve* algorithm.

(*b*) *Comparison between AMR and Uniform Grid*

The next set of calculations compares the composite grid solution with the uniform fine grid solution for a two-phase, three-component system representing a simple polymer flood example. We have an aqueous phase consisting of two components, water (n_w) and polymer (n_p), that is injected into a domain filled with an oleic phase composed of a single component (n_o). We impose the same type of boundary conditions as in the previous two subsections. For this problem, we have $\rho_w = \rho_p = 1000$ kg/m^3, $\mu_{\text{aqueous}} = 0.175 + 0.35(\frac{n_2}{n_1+n_2})$ cP, $\rho_o = 800$ kg/m^3, and $\mu_{\text{oleic}} = 0.35$ cP. The porosity is again uniformly 1. The permeability function $\kappa(x, y)$ is given by (4.3) and the injection composition is given by

$$n_w = \begin{cases} 100 \text{ kg/m}^3, & 0 < t < 5 \times 10^5 \text{s} \\ 900 \text{ kg/m}^3, & t \geq 5 \times 10^5 \text{ s} \end{cases} ;$$

$n_p = 1000 - n_1$ kg/m^3, $t > 0$; and $n_o = 0$ kg/m^3, $t > 0$. Again, the flow is not in the asymptotically second-order regime of the algorithm. The goal here is to show that with the adaptive algorithm one can achieve accuracy comparable to that on a uniform fine grid. The regridding criterion, which flags coarse grid cells for refinement, is based on the magnitude of $|\nabla n_w|$.

We note that although the composition for $t < 5 \times 10^5$ s will lead to a stable front, the composition for $t \geq 5 \times 10^5$ s will lead to a highly unstable front. A direct comparison between the solution after $t = 5 \times 10^5$ s is then not meaningful for reasons elucidated in the previous section. It nevertheless allows us to examine the behavior of both propagation modes of the aqueous phase.

For the uniform grid, we use a grid size of 1024×256. For the AMR simulation we have a base grid of 256×64, with two additional levels of refinement with a factor of two refinement between each level. A particular cell is tagged for refinement if $|\nabla n_w| > 500$. For both simulations, the time step is computed with a CFL number of 0.4. Figure 4 shows that stable solution fronts for the uniform and the adaptive simulations are identical. However, while the unstable fronts exhibit similar features, there are notable differences in the fingering structure. Unstable fronts can thus develop differently even though captured at the same effective resolution, as shown by the uniform and adaptive solutions shown in Figure 5.

In Table 6, we compare the computational time needed to solve the current problem on a uniform grid and an adaptive grid. We note that at 7×10^5 s, AMR reduces the computational time by 88%. However, as the percentage of the domain that is refined increases, the performance benefit decreases. At 10^6 s, when the percentage of the domain refined at level 2 increases to 22.5%, the computational savings from using AMR is reduced to 80%, As expected, the efficiency gains from using AMR will be problem-dependent.

(*c*) *A three-dimensional example*

We consider the same three-component two-phase system for a three-dimensional problem. The domain is now given by $[0, 16]$ m $\times$ $[0, 4]$ m $\times [0, 2]$ m. We impose no-flow boundary conditions along the faces with normals in the y and z direction, and a pressure difference of 2 atm in the x-direction. The properties of the components are the same as in Section (b), and porosity is again uniformly 1. The permeability

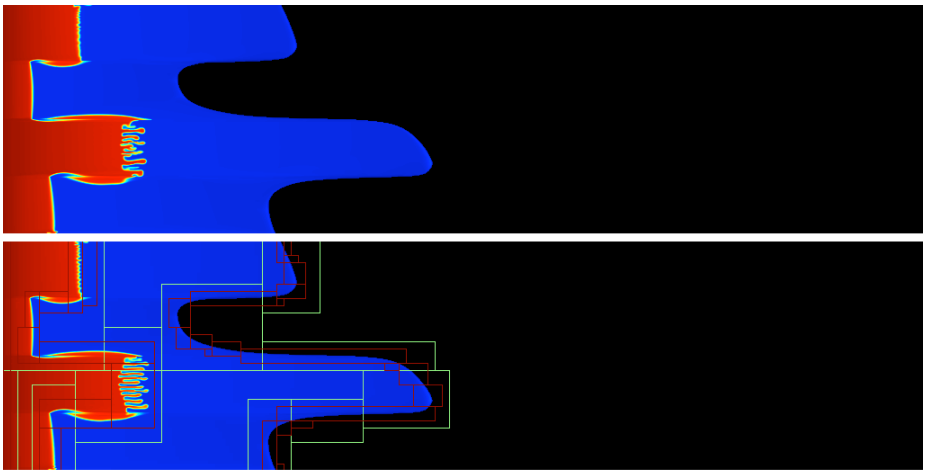

Figure 4. Concentration of water at time 7×10^5 s, solved on an uniform grid (top) and adaptive grid (bottom).

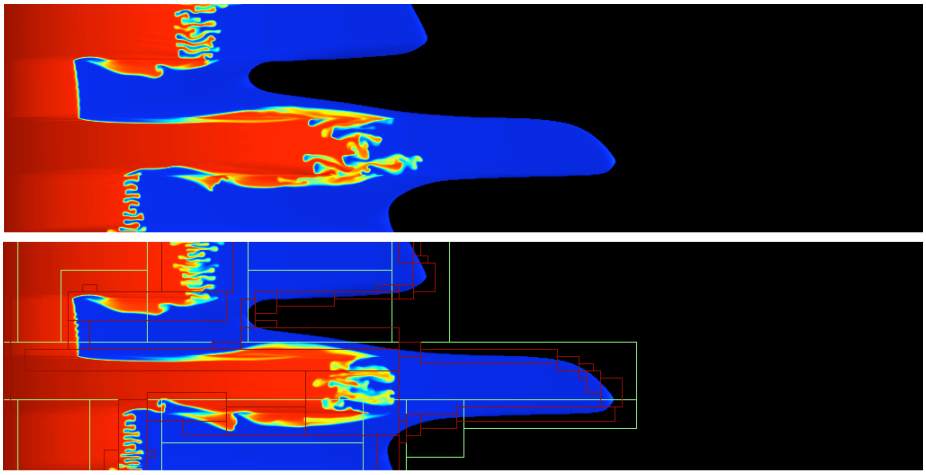

Figure 5. Concentration of water at time 1.04×10^6 s, solved on an uniform grid (top) and adaptive grid (bottom).

Table 6. *Comparison of the computational cost*

	$t = 7 \times 10^5$ s	$t = 1 \times 10^6$ s
time taken for uniform grid	7233.5 s	10461.9 s
time taken for AMR grid	878.4 s	2106.2 s
% domain refined at level 1	26.4	45.6
% domain refined at level 2	11.9	22.5

function $\kappa(x, y, z)$ here is a log-random distribution with a mean of 150 mD and a variance of 0.1 mD. In addition, κ is highly correlated in the x and y directions but not in the z direction, as shown in Figure 6.

We performed the simulation on a Cray XT4 supercomputer. At the coarsest

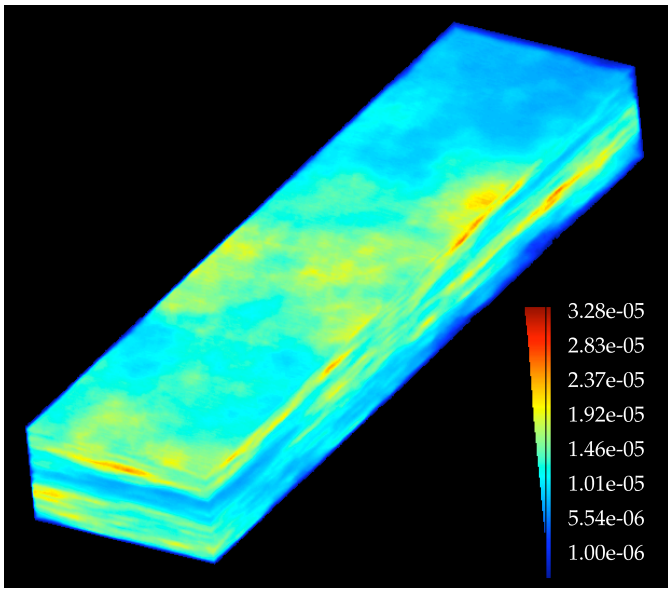

Figure 6. The permeability function used in the 3D example.

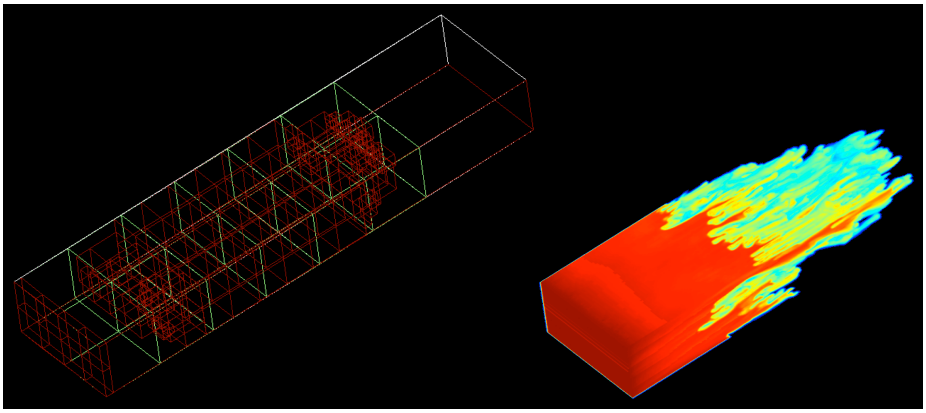

Figure 7. The adaptive grid (left) and solution at time 1.6×10^6 s.

level, the grid size is $128 \times 64 \times 16$ with up to three additional levels of refinement with refinement ratio 2 between adjacent levels. The resulting grid and component density of component water at time 1.6×10^6 s are shown in Figure 7.

(*d*) *Parallel Performance*

By adopting a block-structured form of AMR, the solution at each level in the hierarchy is naturally represented in terms of data defined on a collection of logically rectangular grid patches each containing a large number of points. Thus, the data is represented by a modest collection of relatively large, regular data objects as compared to a point-by-point refinement strategy. This type of approach allows us to amortize the irregular aspects of an adaptive algorithm over large regular

operations on the grid patches. This organization of data into large aggregate grid patches also provides a model for parallelization of the AMR methodology.

Our adaptive methodology is embodied in a hybrid C++/`FORTRAN` software system. In this framework, memory management and flow control are expressed in the C++ portions of the program and the numerically intensive portions of the computation are handled in `FORTRAN`. The software is written using a layered approach, with a foundation library, BoxLib, that is responsible for the basic algorithm domain abstractions at the bottom, and a framework library, AMRLib, that marshals the components of the AMR algorithm, at the top. Support libraries built on BoxLib are used as necessary to implement application components such as interpolation of data between levels, the coarse/fine interface synchronization routines, and linear solvers used in the pressure and diffusion solves.

The fundamental parallel abstraction is the `MultiFab`, which encapsulates the `FORTRAN`-compatible data defined on unions of `Box`s; a `MultiFab` can be used as if it were an array of `FORTRAN`-compatible grids. The grids that make up the `MultiFab` are distributed among the processors, with the implementation assigning grids to processors using the distribution given by the load balance scheme described in Crutchfield (1991) and in Rendleman *et al.* (2000). This load balance scheme is based on a dynamic programming approach for solving the knapsack problem: the computational work in the irregularly sized grids of the AMR data structures is equalized among the available processors. After the initial allocation of grids additional changes to the grid distribution can be performed to reduce communications between processors. For non-reacting flows, the number of cells per grid is often a good work estimate. `MultiFab` operations are performed with an *owner computes* rule with each processor operating independently on its local data. For operations that require data owned by other processors, the `MultiFab` operations are preceded by a data exchange between processors.

Each processor contains *meta-data* that is needed to fully specify the geometry and processor assignments of the `MultiFab`s. At a minimum, this requires the storage of an array of boxes specifying the index space region for each AMR level of refinement. In the parallel implementation, meta-data also includes the processor distribution of the `FORTRAN` compatible data. The meta-data can thus be used to dynamically evaluate the necessary communication patterns for sharing data among processors, enabling us to optimize communications patterns within the algorithm.

To measure the parallel performance of the algorithm, we use a weak scaling study based on a replicated problem strategy as discussed in Colella *et al.* (2007). The case considered here is a two-component single-phase system with a layered permeability function. By replicating the problem in the y and z directions, we are able to scale the problem size without modifying the problem characteristics, particularly with regard to how adaptive criteria and grid generation impact the overall problem. In Figure 8 we present scaling data compared to ideal behavior for a range of processors from 4 to 256. We see a slight deviation from ideal scaling, which is primarily attributable to increases in time spent in the elliptic solver.

5. Conclusions and Future Work

In this paper, we have presented a structured adaptive mesh refinement algorithm for incompressible flows in porous media. The method is based on a total veloc-

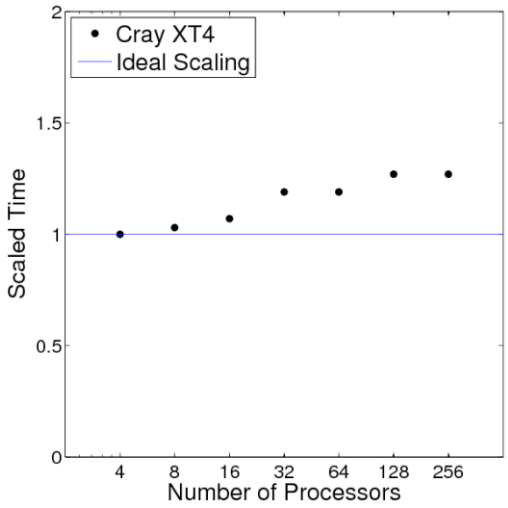

Figure 8. Parallel performance of the adaptive algorithm based on weak scaling.

ity splitting of the equations into an elliptic pressure equation and a system of hyperbolic conservation laws for the component densities. The resulting system is discretized using cell-centered differencing for the pressure equation and a second-order Godunov scheme for the component conservation equations. The adaptive algorithm uses subcycling in time, i.e., the ratio of the time step to the mesh spacing is constant across levels of refinement. Rather than solve a global composite equation for the pressure at every time step at the finest level, we perform single-level solves at each fine step then use an elliptic correction solve to synchronize the solution across levels. This leads to a considerable improvement in computational efficiency. The method has been implemented in parallel and shows excellent scalability up to 256 processors.

Our goal in future work will be to extend this approach to more realistic problems. As a first step in that direction, we will extend the current approach to include realistic geochemical reactions. Beyond that, our target will be to extend the approach to more realistic fluid models that include compressibility, interphase mass transfer and thermal effects.

References

Almgren, A. S., Bell, J. B., Colella, P., Howell, L. H. & Welcome, M. 1998 A conservative adaptive projection method for the variable density incompressible Navier-Stokes equations. *J. Comp. Phys.* **142**, 1–46.

Bell, J. B., Berger, M. J., Saltzman, J. S. & Welcome M. 1994 Three-dimensional adaptive mesh refinement for hyperbolic conservation laws. *SIAM J. Sci. Statist. Comput.* **15**, 127–138.

Bell, J. B., Colella, P. & Trangenstein, J. A. 1989 Higher Order Godunov Methods for General Systems of Hyperbolic Conservation Laws. *J. Comp. Physs.*, **82**, 362–397.

Bell, J. B., Trangenstein J. A. & Shubin, G. R. 1986 Conservation laws of mixed type describing three-phase flow in porous media. *SIAM J. Appl. Math.* **46**(6), 1000–1017.

Berger, M. J. & Colella, P. 1989 Local adaptive mesh refinement for shock hydrodynamics. *J. Comp. Phys.* **82**(1), 64–84.

Berger, M. J. & Rigoutsos, J. 1991 An algorithm for point clustering and grid generation. *IEEE Transactions on Systems, Man, and Cybernetics* **21**, 1278–1286.

Colella, P., Bell, J., Keen, N., Ligocki, T., Lijewski, M. & van Straalen, B. 2007 Performance and Scaling of Locally-Structured Grid Methods for Partial Differential Equations. *J. Phys.: Conf Ser.* **78**, 012013.

Christensen, J., Darche, G., Dechelette, B., Ma, H. & Sammon, P. 2004 Applications of dynamic gridding to thermal simulations. Paper SPE 86969. In *SPE International Thermal Operations and Heavy Oil Symposium and Western Regional Meeting, Bakersfield, California.*

Crutchfield, W. Y. 1991 Load balancing irregular algorithms. Technical Report UCRL-JC-107679, Lawrence Livermore National Laboratory.

Day, M. S. & Bell J. B. 2000 Numerical simulation of laminar reacting flows with complex chemistry. *Combust. Theory Modelling* **4**(4), 535–556.

Edwards, M. G. 1996 A higher-order Godunov scheme coupled with local grid refinement for flow in a porous medium. *Comput. Methods Appli. Mech. Engrg* **131**, 287–308.

Ewing, R. E., Boyett, B. A., Babu, D. K. & Heinemann, R. F. 1989 Efficient use of refined grids for multiphase reservoir simulation. Paper SPE 18413. In *SPE Reservoir Simulation Symposium, Houston, Texas.*

Heinemann, Z. E. 1983 Using local grid refinement in a multiple-application reservoir simulator. Paper SPE 12255. In *SPE Reservoir Simulation Symposium, San Francisco, California.*

Hoang, H. M. & Kleppe, J. 2006 A parallel adaptive finite difference algorithm for reservoir simulation. Paper SPE 99572. In *SPE Europe/EAGE Annual Conference and Exhibition, Vienna, Austria.*

Hornung, R. D. & Trangenstein, J. A. 1997 Adaptive mesh refinement and multilevel iterations for flow in a porous media. *J. Comp. Phys.* **136**, 522–545.

Manickam, O. & Homsy G. M. 1994 Simulation of viscous fingering in miscible displacements with nonmonotonic viscosity profiles. *Physics of Fluids* **6**(1), 95–107.

Nilsson, J., Gerritsen, M. & Younis, R. 2005a An adaptive, high-resolution simulation for steam-injection processes. Paper SPE 93881. In *SPE Western Regional Meeting, Irvine, California.*

Nilsson, J., Gerritsen, M. & Younis R. 2005b A novel adaptive anisotropic grid framework for efficient reservoir simulation. Paper SPE 93243. In *SPE Reservoir Simulation Symposium, Woodlands, Texas.*

Propp, R. M. (1998, Sept.). Numerical Modeling of a Trickle Bed Reactor. Ph.D. thesis, Dept. of Mechanical Engineering, Univ. of California, Berkeley.

Rendleman, C. A., Beckner, V. E., Lijewski, M. J., Crutchfield, W. Y. & Bell J. B. 2000 Parallelization of structured, hierarchical adaptive mesh refinement algorithms. *Computing and Visualization in Science* **3**(3), 147–157.

Sammon, P. H. 2003 Dynamic grid refinement and amalgamation for compositional simulation. Paper SPE 79683. In *SPE Reservoir Simulation Symposium, Houston, Texas.*

Trangenstein, J. A. 2002 Multi-scale iterative techniques and adaptive mesh refinement for flow in porous media. *Advances in Water Resources* **25**, 1175–1213.

Trangenstein, J. A. & Bi, Z. 2002 Large multi-scale iterative techniques and adaptive mesh refinement for miscible displacement simulation. Paper SPE 75232. In *SPE/DOE Improved Oil Recovery Symposium, Tulsa, Oklahoma.*

Xu, T., Sonnenthal, E., Spycher, N. & Pruess K. 2004 TOUGHREACT user's guide: A simulation program for non-isothermal multiphase reactive geochemical transport in variably saturated geologic media. Technical Report LBNL-55460, Lawrence Berkeley National Laboratory.