ICES REPORT 18-03

March 2018

Adaptive Mesh Refinement with an Enhanced Velocity Mixed Finite Element Method on Semi-Structured Grids using a Fully-Coupled Solver

by

Benjamin Ganis, Gergina Pencheva, Mary F. Wheeler

Reference: Benjamin Ganis, Gergina Pencheva, Mary F. Wheeler, "Adaptive Mesh Refinement with an Enhanced Velocity Mixed Finite Element Method on Semi-Structured Grids using a Fully-Coupled Solver," ICES REPORT 18-03, The Institute for Computational Engineering and Sciences, The University of Texas at Austin, March 2018.

Adaptive Mesh Refinement with an Enhanced Velocity Mixed Finite Element Method on Semi-Structured Grids using a Fully-Coupled Solver

Benjamin Ganis[*][†] Gergina Pencheva[*] Mary F. Wheeler[*]

March 26, 2018

Abstract

We describe a novel approach for performing adaptive mesh refinement using mixed finite elements for flow in porous media applications. The enhanced velocity (EV) mixed finite element method is used to construct a strongly flux-continuous velocity approximation between non-matching subdomain grids. In this work, the original EV implementation was generalized to allow interfaces in the interior of subdomains adjacent to inactive cells and to allow dynamic adaptive mesh refinement (AMR). In the new implementation, subdomains with different spatial resolutions are stacked on top of each other to produce a very general semi-structured grid. The decomposition is non-overlapping, but now the subdomains can have holes, have ragged edges, and be nested within each other. Several examples with adaptive mesh refinement are demonstrated in two and three spatial dimensions; *a priori* indicators are used to adapt the grid for a multiphase compositional flow model, and *a posteriori* indicators are used to adapt the grid for a single phase flow model. Moreover, a new fully-coupled linear solver for the EV method is also implemented in this work, which shows a dramatic reduction in the number of Newton iterations versus the previously implemented nonlinear block Jacobi solver, especially when systems have a strong elliptic component.

Keywords. adaptive mesh refinement; mixed finite elements; enhanced velocity method; semi-structured grid; block-structured AMR, fully-coupled solver

1 Introduction

The enhanced velocity (EV) mixed finite element method was first introduced in [35] as a way to model flow in porous media on non-matching multiblock grids. The basic idea is that a fluid model is discretized using standard mixed finite element methods such as Raviart-Thomas (RT) elements [30, 26] on non-overlapping subdomains. These subdomains have typically have structured Cartesian grids, but their traces are allowed to be spatially non-conforming at subdomain interfaces. Extra degrees of freedom for the normal fluxes are added (i.e., the velocity is enhanced) along the subfaces of the intersection mesh, giving exact normal flux continuity between subdomains by construction. This method is known to achieve element-wise local mass conservation throughout the entire domain, including at spatial non-conformities, and it has been shown to be a convergent scheme.

[*]Center for Subsurface Modeling, The Institute for Computational Engineering and Sciences (ICES), The University of Texas at Austin, Austin, Texas, USA.

[†]Corresponding author, `bganis@ices.utexas.edu`.

The novelty of this paper describes how we have extended the EV mixed finite element method in several important ways:

1. We pose the EV method on semi-structured grids such that the non-overlapping subdomains may have holes, have ragged edges, and be nested within each other.
2. We incorporate adaptive mesh refinement (AMR) so that the spatial decomposition changes dynamically at each time step.
3. We use a fully-coupled linear solver in the Newton iteration that is an improvement over the previously used nonlinear block Jacobi solver.

Our parallel implementation in the IPARS (Integrated Parallel Accurate Reservoir Simulator) code is demonstrated on both single phase slightly compressible flow and fully compositional multiphase flow models. Figure 1 shows an admissible semi-structured grid that is possible with these new extensions. Each subdomain is composed of a set of active elements from underlying Cartesian grids that cover the entire domain. They are numbered such that each subsequent subdomain has a finer mesh resolution and its underlying grid is evenly nested the preceding level. The union of all subdomains covers the domain in a non-overlapping fashion. This grid can now change at every time step.

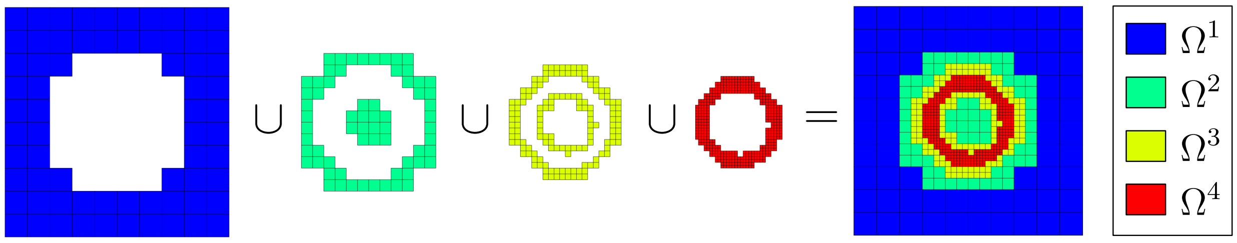


Figure 1: An example of a semi-structured grid comprised of the union of four non-overlapping polygonal subdomains with different spatial resolutions.

We note that our concept of a semi-structured EV grid is more general than quadtree or octree meshes [17], which are tree-based data structures that have been used in scientific computation since the 1970's for nonconforming local refinements of structured grids. The EV method is more general because this mixed finite element technique is well-posed on grids with completely arbitrary spatial non-conformities. Therefore, the refinement ratios between any two levels (i.e. subdomains) in each spatial dimension need not be restricted to two. In fact, there may be different refinement ratios between any two given grid levels. Moreover, an element of one refinement level may be adjacent to any element of any other refinement level, so the meshes need not be graded, although doing so gives a smoother solution. However in spite of this supported generality, the types of grids used in the presented numerical examples are indeed special cases of quadtree and octree meshes, upon which we pose mixed finite element methods to solve single and multiphase flow in porous media.

In order to drive any AMR procedure, it is necessary to have indicators that mark areas of the domain for refinement or de-refinement. In practice this is done through the calculation of *a posteriori* error indicators. In order to meet the aforementioned three goals, it is beyond the scope of this paper to develop new *a posteriori* error indicators. We will therefore take the following approach: As a proof of concept for the complex physics arising from a multiphase flow model, we will employ *a priori* error indicators that determine grid refinement through advance knowledge of the fine scale solution. For a simpler single phase flow model, we will employ a residual-based *a posteriori* error indicator. For the necessary details of the fully compositional equation-of-state multiphase flow model, the reader is referred to the recent work of [20] and the references therein for the appropriate formulation using a two-point flux cell-centered finite difference scheme. The compositional model is quite complicated, but since the model itself is not under study in this paper, we omit the details for brevity. We reiterate that the focus of this paper merely on the construction of the EV method on a new type of dynamic grid with a fully-coupled solver, which is applicable to both a variety of flow models and *a posteriori* error indicators.

The analysis of *a posteriori* error indicators in the context of AMR is an active area of research with a rich history that is highly dependent on the partial differential equations and finite element methods under consideration. The textbooks [2] and [6] and their many references give a broad overview of indicators and refinement strategies for a variety of applications and numerical schemes. With respect to AMR applied to mixed finite element methods, we mention some specific works here. In [24], a steady-state elliptic problem was studied with mixed hybrid finite elements, which considered nonconforming refinements of rectangular Cartesian grids, as well as both conforming and nonconforming local refinements of unstructured triangulations in two dimensions. In [7], a Richard's equation for partially saturated flow was considered with RT mixed finite elements on conforming refinements of unstructured triangles. In [12], an analysis was performed using RT mixed finite elements for an elliptic problem with homogeneous coefficients. In [37], the multiscale mortar mixed finite element method was used to construct *a posteriori* error indicators for an elliptic problem, whereby local refinements were utilized on a subdomain level. In [28], alternative *a posteriori* indicators were constructed for the mortar method based on post-processing techniques to give efficiency, as well as guaranteed upper and lower bounds on the errors. With respect to recent work on quadtree and octree implementations, we note the useful p4est library [11], although it is not employed in this work. An example of its utility is demonstrated in the recent work [15] on AMR for a non-porous media two-phase equation-of-state flow for combustion applications.

We would also like to point out the many generalizations and extensions of the underlying mixed finite element techniques that have been developed over many decades. The relationship between the lowest order Raviart-Thomas (RT_0) mixed finite element spaces on structured Cartesian grids and the standard two-point flux cell-centered finite difference method (CCFD) has been established since [31]. Here, CCFD was shown to be a special case of RT_0 through the application of a special quadrature rule. The so-called expanded mixed finite element methods have been developed to support full tensor permeability coefficients [5]. Mortar methods have been used to couple subdomains with Lagrange multipliers since the seminal work of [23], followed up by a thorough multiscale analysis in [4]. The EV method was formulated as an alternative to the mortar method in [35] with an implementation based on a nonlinear block Jacobi solver. The mortar method was originally formulated as an iterative interface problem, interface preconditioners were developed [29], and the algorithm was also modified through the formulation of a multiscale basis [22]. The mortar method was reformulated again as a fully-coupled global Jacobian system [18], which made this computational speed comparable to what was observed for the EV method. This fully-coupled construction is done for the EV method in this paper for further improvement. More recently, similar velocity elimination techniques have been used on distorted quadrilateral and hexahedral grids, whereby other quadrature techniques have been applied to the BDM [9] or BDDF [8] mixed finite element spaces in order to formulate the multipoint flux mixed finite element methods (MFMFE) [36]. The MFMFE method has a strong similarity to the MPFA-O method [1], with the added benefit of having a variational formulation enabling rigorous convergence analysis. Continuting work is extending the MFMFE method to nonconforming hexhedral grids using EV techniques [21] as well as local flux [25] techniques. All of these methods began with a mathematical formulation on a simple single phase flow model, but were then generalized more complicated multiphase flow models as in [14, 19, 20]. Only modest success was made in adaptive mesh refinement, since subdomains were statically placed in separate spatial locations. In this paper, the simple reconsideration of these methods with semi-structured grids demonstrates that much more powerful AMR is possible in this extensive body of work.

The format of our paper is as follows. In Section 2, we review the formulation of the EV method and describe details of its implementation on semi-structured grids. In Section 3, we describe the algorithms used for adaptive mesh refinement. In Section 4, we describe the fully-coupled linear solver and its differences over the nonlinear block Jacobi solver. In Section 5, we give a variety of numerical examples for two and three dimensions, single phase and compositional flow models, and *a priori* and *a posteriori* based grid adaptivity. These adaptive results show a significant speedup over a fine grid solution, while maintaining critical solution accuracy where it is needed. In Section 6, we give some concluding remarks.

2 The EV Method on Semi-Structured Grids

In this section, we review the formulation of the EV method for flow in porous media on non-matching multiblock domains with a slightly compressible single phase flow model, similar to the presentation in [34]. We note that the EV method has been implemented for several multiphase flow models including fully compositional flow, but for simplicity in the presentation we describe it for a single phase model. The novelty in this work is the extension of the EV method to dynamic semi-structured grids.

Consider a time interval $[0, T]$, along with a spatial domain, $\Omega \subset \mathbb{R}^d$, for $d = 2$ or 3, with boundary $\partial\Omega$ and outward normal $\mathbf{n}$. We assume the spatial domain is polygonal with edges parallel to the Cartesian coordinate axes. It is decomposed into N non-overlapping subdomains such that

$$\overline{\Omega} = \bigcup_{k=1}^{N} \overline{\Omega_k}, \qquad \Omega_k \cap \Omega_l = \emptyset \text{ when } k \neq l. \tag{1}$$

Pairs of subdomain interfaces are denoted by $\Gamma_{kl} = \Gamma_{lk} = \partial\Omega_k \cap \partial\Omega_l$, the union of all interfaces are denoted by $\Gamma = \bigcup_{1\leq k<l\leq N_\Omega} \Gamma_{kl}$, and interior subdomain interfaces are denoted by $\Gamma_k = \partial\Omega_k \cap \Gamma = \partial\Omega_k \setminus \partial\Omega$. We assume the subdomains Ω_k have the same polygonal shape restriction as the domain, but note they may have holes and be nested within each other. Furthermore, a single subdomain may also be comprised of a union of non-contiguous polyhedra. To incorporate adaptive mesh refinement, the spatial decomposition is allowed to change dynamically, but since it is always a non-overlapping, the decomposition satisfies the property (1) at each point in time.

The model partial differential equations for slightly compressible single-phase flow through a porous medium are given by

$$\mathbf{u} = -\frac{K}{\mu}\rho\left(\nabla p - \rho\mathbf{g}\right) \qquad \text{in } \Omega \times (0, T], \tag{2a}$$

$$\frac{\partial}{\partial t}(\phi\rho) + \nabla\cdot\mathbf{u} = q \qquad \text{in } \Omega \times (0, T], \tag{2b}$$

$$\mathbf{u}\cdot\mathbf{n} = 0 \qquad \text{on } \partial\Omega \times (0, T], \tag{2c}$$

$$p = p_0 \qquad \text{on } \Omega \times \{t = 0\}. \tag{2d}$$

The subsurface flow is characterized by Darcy's law (2a) together with a conservation of mass (2b). The primary unknowns are $p(\mathbf{x}, t)$ for fluid pressure and $\mathbf{u}(\mathbf{x}, t)$ for the Darcy velocity. Given data include the porosity $\phi(\mathbf{x})$, the mass flux source term $q(\mathbf{x}, t)$, the second order permeability tensor $K(\mathbf{x})$, the fluid viscosity μ, and the gravitational acceleration vector $\mathbf{g}$. No-flow boundary conditions are assumed in (2c), but we note the extension to more general boundary conditions is straightforward. The initial condition (2d) is assumed to be under hydrostatic equilibrium. The density, $\rho = \rho(p)$, satisfies the slightly compressible equation of state,

$$\rho = \rho^{\text{ref}} e^{c(p-p^{\text{ref}})}, \tag{3}$$

where ρ^{ref} is the reference density at the reference pressure p^{ref}, and c is the fluid compressibility. The nonlinearity of this model is mild, owing to the fact that c is small. The permeability tensor K is assumed to be uniformly bounded and positive definite, i.e., there exist constants $0 < k_0 \leq k_1 < \infty$ such that

$$k_0 \|\xi\|^2 \leq \xi^T K(\boldsymbol{x}) \xi \leq k_1 \|\xi\|^2 \quad \forall \xi \in \mathbb{R}^d, \ \forall \boldsymbol{x} \in \Omega, \tag{4}$$

where $\|\cdot\|$ is the Euclidean vector norm.

The functional spaces for the dual mixed weak formulation [10] of (2) are $\mathbf{V} = H(\text{div}; \Omega) = \{\mathbf{v} \in (L^2(\Omega))^d : \nabla \cdot \mathbf{v} \in L^2(\Omega)\}$ and $W = L^2(\Omega)$. The weak solution is a pair $\mathbf{u} \in \mathbf{V}$, $p \in W$ such that

$$(\mu \rho^{-1} K^{-1} \mathbf{u}, \mathbf{v}) - (p, \nabla \cdot \mathbf{v}) - (\rho \mathbf{g}, \mathbf{v}) = 0, \qquad \forall \mathbf{v} \in \mathbf{V}, \tag{5a}$$

$$\left(\frac{\partial(\phi\rho)}{\partial t}, w \right) + (\nabla \cdot \mathbf{u}, w) = (q, w), \qquad \forall w \in W, \tag{5b}$$

where $(\cdot, \cdot)$ denotes the $L^2(\Omega)$-inner product.

2.1 Fully-Discrete Formulation with EV

To write the fully-discrete finite element discretization of (5a)–(5b), we first break the temporal domain into discrete time steps $0 = t_0 < t_1 < \ldots < t_M = T$, such that $\Delta t_m = t_m - t_{m-1}$, for which we will employ the implicit backward Euler method. At each time step m, the non-overlapping domain decomposition may change, so we add the subscript m to denote the subdomain $\Omega_k = \Omega_{k,m}$ at $t = t_m$.

The spatial discretization uses the Raviart–Thomas mixed finite element spaces of lowest order RT_0 [30, 26] on rectangles (if $d = 2$) or bricks (if $d = 3$). Let $\mathcal{T}_{h,k}$ be a static tensor-product Cartesian grid that covers $\overline{\Omega}$ with mesh resolution h_k. Let $\mathcal{T}_{h,m,k}$ be the dynamic subdomain grid that is related to the static mesh $\mathcal{T}_{h,k}$, whereby its elements are either active or inactive at time level m. We associate the global characteristic mesh width $h = \max_{k=1}^{N}(h_k)$ in terms of the subdomain with the coarsest resolution, and we order the subdomains $k = 1, \ldots, N$ from coarsest to finest. At time level m, the global mesh is denoted $\mathcal{T}_{h,m} = \bigcup_{k=1}^{N} \mathcal{T}_{h,m,k}$. The traces of subdomain grids are spatially non-conforming at the dynamic interface Γ_m, and the dynamic discretization is non-overlapping and completely covers $\overline{\Omega}$.

Let $\{x_1, \ldots, x_d\}$ denote an orthonormal coordinate system in a frame of reference. For each active Cartesian element $E \in \mathcal{T}_{h,m}$, the RT_0 spaces are defined as follows:

$$\begin{aligned} \mathbf{V}_{h,m}(E) &= \{\mathbf{v} = (v_1, \ldots, v_d)^T : v_i = a_i + b_i x_i; \ a_i, b_i \in \mathbb{R}, \ i = 1, \ldots, d\} \\ W_{h,m}(E) &= \{w = c : c \in \mathbb{R}\}. \end{aligned}$$

On interior edges of subdomains and external boundaries, the velocity $\mathbf{v} \in \mathbf{V}_{h,m}(E)$ is uniquely determined by degrees of freedom corresponding to the normal components $\mathbf{v} \cdot \mathbf{n}$ at midpoints of edges (or faces) of E, where $\mathbf{n}$ is the outward unit normal of ∂E. These normal flux degrees of freedom are shared between neighboring elements in the same domain. The degree of freedom uniquely determining the pressure $w \in W_{h,m}(E)$ is a single constant corresponding to its value at the center of E.

The basic idea of the enhanced velocity space $\mathbf{V}^*_{h,m} \subset \mathbf{V}$ is to introduce extra velocity degrees of freedom on spatially non-conforming interfaces, in order to strongly enforce global mass conservation. This is constructed such that one face centered normal flux degree of freedom exists on each sub-face between any two neighboring elements belonging to neighboring finite element spaces of different subdomains. See Figure 2 for diagrams for the degrees of freedom in both the RT_0 and EV spaces. In this picture, the coarse element adjacent to the non-conforming boundary has two

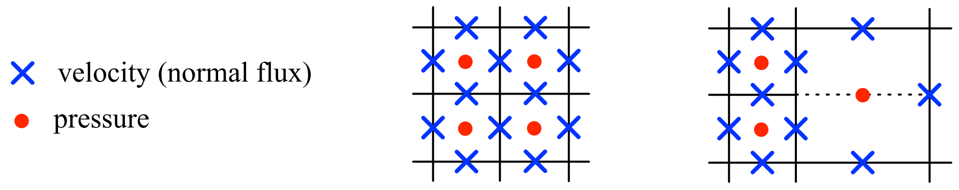


Figure 2: Degrees of freedom for velocity and pressure of the RT_0 space for $d = 2$ on rectangles (left) and the enhanced velocity space (right) near a non-conforming interface.

normal flux degrees of freedom on its left face, instead of the usual one degree of freedom per face for elements on the strict interior of a subdomain mesh.

The formal definitions of the global finite element spaces are defined as follows. At each time step, the global discrete pressure space is defined by

$$W_{h,m} = \{w \in L^2(\Omega) : w|_E \in W_{h,m}(E),\ \forall E \in \mathcal{T}_{h,m}\}.$$

No modifications are necessary to the pressure space of the global domain near non-conforming interfaces, because it is comprised of discontinuous constants by construction. For each subdomain $k = 1, \ldots, N$, the usual RT_0 velocity spaces are

$$\mathbf{V}_{h,m,k} = \{\mathbf{v} \in H(\mathrm{div}; \Omega_{k,m}) : \mathbf{v}|_E \in \mathbf{V}_{h,m}(E),\ \forall E \in \mathcal{T}_{h,m,k},\ \text{and } \mathbf{v} \cdot \mathbf{n} = 0 \text{ on } \partial\Omega\}.$$

As described in [35], the direct sum of these subdomain velocity spaces is not a finite dimensional subspace of $H(\mathrm{div}; \Omega)$, because the normal vector components do not match at the spatial non-conformities on Γ_m. To construct the EV space, first define $\mathbf{V}^0_{h,m,k} = \{\mathbf{v} \in \mathbf{V}_{h,m,k} : \mathbf{v} \cdot \mathbf{n} = 0 \text{ on } \Gamma_k\}$ to remove velocity degrees of freedom along the interface. Next discretize the interface into a regular partition $\mathcal{T}^{\Gamma}_{h,m}$ equal to the intersection of the traces of the non-conforming subdomain grids. When subdomain grids have an evenly nested relationship, as employed in this work, the interface grid $\mathcal{T}^{\Gamma}_{h,m}$ is a subset of existing subdomain faces[1]. For any subdomain element $E \in \mathcal{T}_{h,m,k}$ adjacent to interface Γ_m, the interface grid $\mathcal{T}^{\Gamma}_{h,m}$ may divide the boundary face into several parts, which can subdivide the element E into several pieces, see e.g. the dashed line in Figure 2. For each of these sub-elements, we define RT_0 basis functions with normal component equal one on the interface edge and zero on all other edges. We then define $\mathbf{V}^{\Gamma}_{h,m}$ to be the span of all such basis functions. Finally, the global discrete enhanced velocity space is defined as

$$\mathbf{V}^*_{h,m} = (\mathbf{V}^0_{h,m,1} \oplus \cdots \oplus \mathbf{V}^0_{h,m,N} \oplus \mathbf{V}^{\Gamma}_{h,m}) \cap H(\mathrm{div}; \Omega).$$

[1]In general, the EV method can handle completely non-matching interfaces in which new face subdivisions are introduced that do not belong to existing traces of subdomain grids, e.g. Figure 3 left.

For time steps $m = 1, \ldots, M$, the fully-discrete solution is to find a sequence of pairs $\mathbf{u}_{h,m} \in \mathbf{V}^*_{h,m}$, $p_{h,m} \in W_{h,m}$ such that

$$(\mu \rho_{h,m}^{-1} K^{-1} \mathbf{u}_{h,m}, \mathbf{v}_h) - (p_{h,m}, \nabla \cdot \mathbf{v}_h) - (\rho_{h,m} \mathbf{g}, \mathbf{v}_h) = 0, \qquad \forall \mathbf{v}_h \in \mathbf{V}^*_{h,m}, \tag{6a}$$

$$\left(\frac{(\phi\rho)_{h,m} - \mathcal{P}_m (\phi\rho)_{h,m-1}}{\Delta t_m}, w_h \right) + (\nabla \cdot \mathbf{u}_{h,m}, w_h) = (q, w_h), \qquad \forall w_h \in W_{h,m}, \tag{6b}$$

where $\mathcal{P}_m : W_{h,m-1} \to W_{h,m}$ is a mass-preserving L^2 projection operator that projects a piecewise constant finite element function onto the adapted grid of the subsequent time level. Fluid compressibility $c > 0$ leads to a nonlinear system, so it will be solved incrementally using Newton's method on each time step. In matrix form, let us call the fully discrete linear system for a Newton step

$$\begin{bmatrix} A & B \\ \Delta t B^T & D \end{bmatrix} \begin{bmatrix} \delta U \\ \delta P \end{bmatrix} = - \begin{bmatrix} F \\ G \end{bmatrix}. \tag{7}$$

For more details, this matrix notation follows the paper [18], except here we do not introduce Lagrange multipliers into the linear system as in the multiscale mortar method. Residual equations (6a) and (6b) correspond to vectors F and G respectively, and its Jacobian matrix of partial derivatives gives the structure of matrix blocks A, B, and D.

Velocity elimination is performed on the matrix system for the EV mixed finite element method. The first term in (6a) is evaluated using a trapezoidal-midpoint quadrature rule [31], which diagonalizes the A matrix. This gives a straightforward way to convert the indefinite saddle point system into a non-symmetric positive definite system in terms of the pressure increment alone.

$$\underbrace{\left(\Delta t \; B^T A^{-1} B - D\right)}_{J} \delta P = \underbrace{\left(G - \Delta t \; B^T A^{-1} F\right)}_{-R} \tag{8}$$

The special case of the EV mixed finite element method with special quadrature and velocity elimination yields a Jacobian matrix J and residual vector R that is equivalent to a cell-centered finite difference (CCFD) scheme. In areas of the domain where the grid is spatially conforming, this gives a standard two-point flux scheme with a 5-point or 7-point stencil when $d = 2$ or $d = 3$, respectively. For areas of spatial non-conformity, this gives a two-point flux scheme with unstructured matrix blocks, due to unstructured nature of the the enhanced velocity degrees of freedom $\mathbf{V}^\Gamma_{h,m}$, in which one coarse element may interact with several finer elements through a single face.

2.2 Extension to Semi-Structured Grids

In the conventional EV method that has typically been used from its initial development [35] until present, three restrictions were present in the IPARS code framework: (1) interfaces were restricted to be on the external boundaries of subdomains, (2) subdomains could not be nested within each other, and (3) the decompositions were static in time.[2] These restrictions are somewhat artificial, because they have nothing to do with the EV method itself, but rather the design choices made in the initial code implementation.

Figure 3 shows two typical decompositions that were possible under the previous implementation. The left figure shows three structured subdomains with different resolutions coming together in a spatially non-conforming way. Due to the restrictions placed upon the code framework, the

[2]We note that restriction (3) has also been relaxed in the concurrent work of [3, 33] in a separate Matlab research code.

subdomains had to be placed "side-by-side". This meant when local refinements were needed in the interior of the domain, it needed to be broken up into many logically structured pieces. The right figure shows five subdomains were necessary to handle a single area of mesh refinement. Due to these artificial restrictions, if many areas of local refinement were needed, the number of subdomains necessary in the previous implementation quickly became untenable.

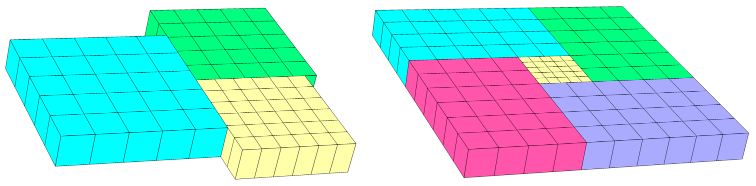


Figure 3: Two examples of decompositions possible under the previous EV implementation where interfaces were restricted to external subdomain boundaries. Colors represent subdomains.

In this work, we have made changes to the code framework to relax the above three arbitrary assumptions that were placed upon the IPARS code framework. More specifically, structured subdomain grids are now allowed to have inactive cells, such that other subdomains may have active cells in these vacant areas, see Figure 1. From this point of view, the subdomains are placed "on top" of each other, and each subdomain is associated with its own mesh refinement level. We define this as a **semi-structured grid**. This enables adaptive grid refinement in any part of the domain with a minimal number of subdomains and greater ease of use. It also allows the method to accurately resolve complex solution features such as moving fronts and domain features such as fractures and channels. These refinement techniques can be driven with error estimation.

The refinement ratios between any two consecutively numbered subdomains r_k, for $k = 1, \ldots, N-1$, are defined such that if Ω_k has a $n_1 \times \ldots n_d$ grid, then Ω_{k+1} has a $r_k \cdot n_1 \times \ldots r_k \cdot n_d$ grid. In the special case when $r_k = 2$, for $k = 1, \ldots, N-1$, this construction is similar to a quadtree or octree mesh [17] in two or three spatial dimensions. However, the semi-structured grids in the EV construction are more general than quadtree meshes in several ways: (1) the refinement ratios can be any integer $r_k \geq 2$, (2) the refinement ratios may be different such that $r_k \neq r_l$ is allowed for $k \neq l$, and (3) the meshes need not be graded, as $\Gamma_{kl} \neq \emptyset$ is allowed for levels $|k - l| > 2$. These generalizations are possible, because the EV method can easily handle any type of arbitrary spatial non-conformity between elements.

Recall Figure 1 that shows an example of an admissible spatial decomposition using a semi-structured grid with four refinement levels. This grid has been refined to resolve the sharp gradient of a saturation front caused by a radial flow pattern around a fluid injection source. The first subdomain Ω_1 has a 10×10 grid, and the others are refined by a factor of 2 in each dimension, leading to Ω_4 having a 80×80 grid. Each subdomain contains inactive cells (not shown) to create vacant areas where other subdomains may have active regions. The subdomains are polyhedral in shape but not necessarily contiguous, as observed in Ω_2 and Ω_3. The mesh was graded so that subdomain Ω_k is only adjacent to subdomains Ω_{k-1} or Ω_{k+1}, but note this is not a requirement by the EV method. We reiterate that property (1) is always satisfied.

3 Adaptive Mesh Refinement Algorithms

The main goals of this paper are to generalize the EV method to semi-structured grids, to demonstrate dynamic adaptive mesh refinement, and to improve the solver of the resulting systems.

However, in order to drive any dynamic grid refinement procedure, it is necessary to utilize some type of error indicator. As discussed in Section 1, the literature is extensive but oftentimes highly specific to physical models and numerical methods. Please note that it is beyond the scope of this paper to formulate new error indicators; the numerical examples serve as proofs of concept for the stated goals. These examples employ two types of error indicators: *a priori* and *a posteriori*. The former assumes prior knowledge of a fine scale solution and the latter does not. Figure 4 shows high-level flow charts for the two types of refinement algorithms.

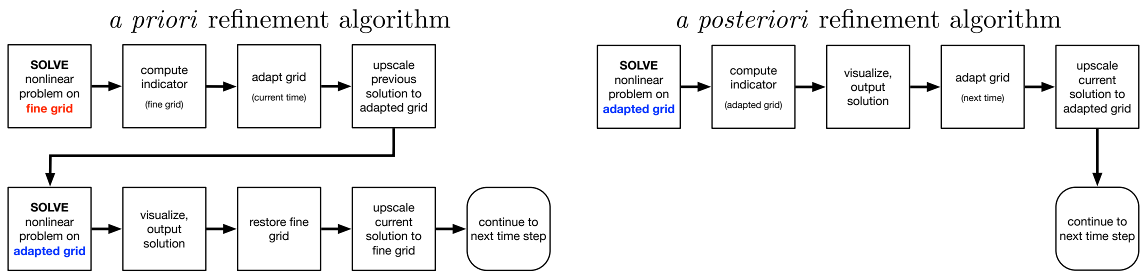

a priori refinement algorithm
SOLVE nonlinear problem on fine grid
compute indicator (fine grid)
adapt grid (current time)
upscale previous solution to adapted grid
SOLVE nonlinear problem on adapted grid
visualize, output solution
restore fine grid
upscale current solution to fine grid
continue to next time step
a posteriori refinement algorithm
SOLVE nonlinear problem on adapted grid
compute indicator (adapted grid)
visualize, output solution
adapt grid (next time)
upscale current solution to adapted grid
continue to next time step

Figure 4: Flow charts for the two types of refinement algorithms used in this work.

In the *a priori* refinement algorithm given in Figure 4 (Left), on any given time step, we first solve a fine scale problem.[3] The fine scale solution is used to compute one or more indicator functions. At locations where indicator functions exceed given thresholds, elements are marked to be on the finest level, and the mesh is graded to be gradually coarsened in unmarked regions. After constructing this adaptive grid, the solution from the previous time step is projected onto the adapted grid, and the nonlinear problem is resolved on the same time step. The adaptive solution is projected back onto the fine grid in preparation for the next time step. In this way, the fine scale solution was only determined to construct the indicator itself.

In the *a posteriori* refinement algorithm given in Figure 4 (Right), we start with a current adapted grid that was determined from the previous time step, and solve the nonlinear problem. An error indicator is computed on the current adapted grid in order to mark cells that exceed refinement or de-refinement thresholds. From one adaptive grid to the next, an element may move up or down one refinement level. This procedure could potentially be iteratively performed several times within a given time step, but in this work it is only performed once. Hence, indicators computed on the current adapted grid will determine the adapted grid of the subsequent time step. Unlike the previous refinement algorithm, this is the type of method that should drive a realistic AMR procedure when it is employed in practice.

Algorithmic flow charts for marking cells from indicator functions and constructing the adaptive grids on each time step are given in Figure 5. For parallel considerations, places where interprocess communication is necessary to populate ghost layers are marked UPDATE. Note the *a priori* case starts with an indicator on the fine grid and marks cells to belong to the finest level where thresholds are exceeded. Areas of fine grid are completely determined by the computation of the current indicator, and are independent of adaptive grid used at the previous time step. In contrast, the *a posteriori* case starts with an indicator on the current adaptive grid, and makes decisions on how to refine or de-refine individual cells from one adaptive solution to the next. Each algorithm finishes with a loop to grade the mesh, to improve solution quality by enforcing smooth changes in

[3]Please note, one would never employ the *a priori* refinement method in practice, because it is computationally more costly than the fine scale problem, while giving a less accurate solution. Nevertheless, it serves an important purpose: to demonstrate the quality and efficiency of the EV solution on a predetermined adaptive semi-structured grid, while decoupling it from the complex issue of finding a good *a posteriori* indicator.

refinement levels between neighboring cells.

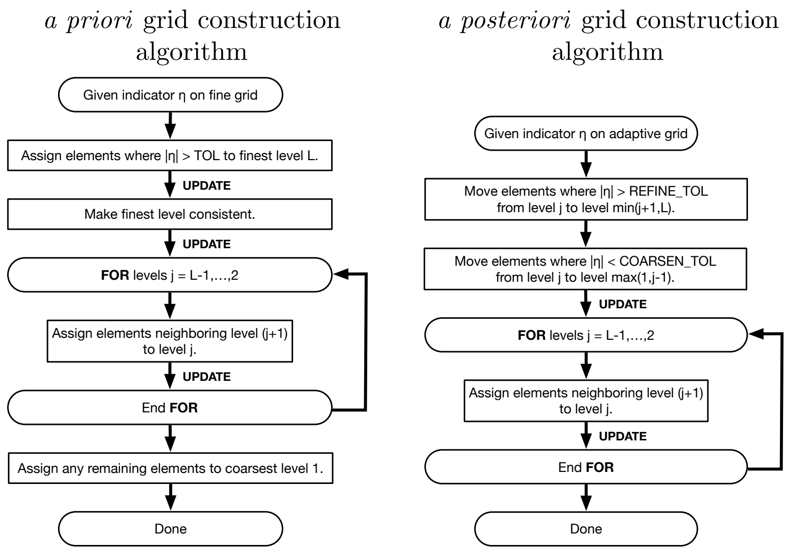

a priori grid construction algorithm
Given indicator η on fine grid
Assign elements where |η| > TOL to finest level L.
UPDATE
Make finest level consistent.
UPDATE
FOR levels j = L-1,...,2
Assign elements neighboring level (j+1) to level j.
UPDATE
End FOR
Assign any remaining elements to coarsest level 1.
Done
a posteriori grid construction algorithm
Given indicator η on adaptive grid
Move elements where |η| > REFINE_TOL from level j to level min(j+1,L).
Move elements where |η| < COARSEN_TOL from level j to level max(1,j-1).
UPDATE
FOR levels j = L-1,...,2
Assign elements neighboring level (j+1) to level j.
UPDATE
End FOR
Done

Figure 5: Flow charts for adaptive grid construction for the two refinement techniques.

There are several other considerations that must be taken into account in the reservoir simulator when the grids are adaptively refined. All state variables must be appropriately upscaled/downscaled using the L^2-projection operator $\mathcal{P}_m$ from one grid to the next. In multiphase equation of state compositional models, flash calculations must then be reinitialized to maintain mass conservation between components. Since grid adaptivity changes the number of degrees of freedom in the finite element method, the linear system must be reordered such that each row corresponds to the current set of active elements. More specifically to the EV method, the transmissibility factors between any two elements that have been adapted must be recalculated.

Details of the L^2-projection $\mathcal{P}_m : W_{h,m-1} \to W_{h,m}$ are as follows. For any scalar function $q_h \in W_{h,m-1}$, the projection operator is defined by

$$\int_\Omega (q_h - \mathcal{P}_m q_h) w = 0, \quad \forall w \in W_{h,m}. \tag{9}$$

The finite element function q_h is a piecewise constant function on the grid $\mathcal{T}_{h,m-1}$ and its projection $\mathcal{P}_m q_h$ is a piecewise constant function on the grid $\mathcal{T}_{h,m}$. Under the assumption that the grids are nested, upscaling and downscaling operations have decoupled local support, so a global linear system is not necessary to solve. Moreover, using midpoint quadrature, (9) reduces to a volume-weighted arithmetic average for the case of upscaling, and a direct copy of coarse to fine values for the case of downscaling. As shown in Figure 6, the upscaling case on the left gives the formula $\overline{x} = \frac{1}{n}\sum_{i=1}^{n} w_i x_i = \frac{1}{16}(x_1 + x_2 + x_3 + x_4) + \frac{1}{4}(x_5 + x_6 + x_7)$, and the downscaling case on the right gives $x_i = \overline{x}$, for $i = 1, \ldots, n$.

Finally, we briefly comment on the parallelism and data structures that we have employed in this work. It is important to note that every element's processor ownership is completely unrelated to its subdomain ownership. In our current implementation, we have made the choice to decompose the global problem to evenly divide the elements in underlying coarsest subdomain $\mathcal{T}_{h,k=1}$ amongst the processors. Since subdomain grids are nested in our semi-structure grid, any regions of local refinement will inherit their processor ownership from their corresponding coarse element in which they belong. This is shown in Figure 7 (left), where colors correspond 8 different processors. This

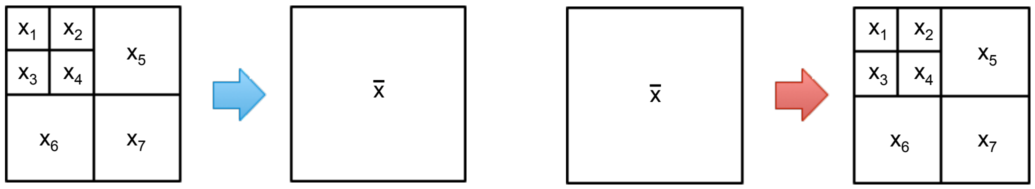


Figure 6: Illustrations of upscaling (left) and downscaling (right) procedures for piecewise constant finite element functions on nested grids.

type of parallel decomposition is the easiest to implement, because no interprocess communication is necessary during the upscaling procedure. However, if areas of local refinement are highly localized to just a few processors, then the computational load may become unbalanced in both matrix assembly and linear solve operations. A more ideal solution would be to attempt to balance the number of elements in the current adaptive grid of every time step, but this would come with its own costs and complications. Several software packages implement a variety of dynamic load balancing techniques for adaptive grids, such as ParMetis [32].

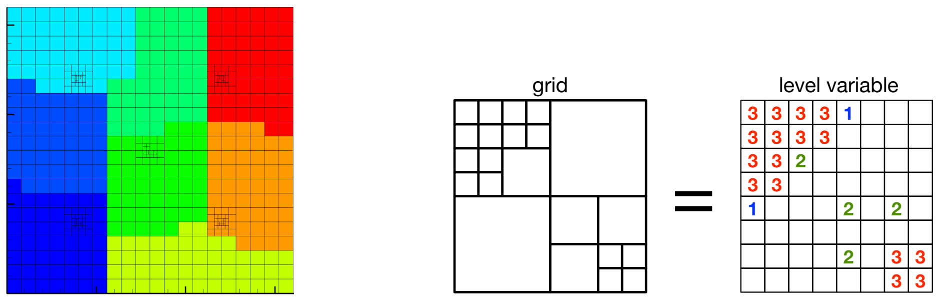


Figure 7: Illustrations of parallel processor decomposition based on the coarsest grid (left) and dense data structures based on the finest grid (right).

Our proof-of-concept of dynamic adaptive mesh refinement using semi-structured grid for the enhanced velocity method was implemented in the legacy research code IPARS. To represent cell-centered scalar data and unknowns, this code currently relies upon dense grid memory allocation for each underlying subdomain that is performed once upon problem initialization. Obviously, this design choice did not anticipate being employed for changing the problem discretization at runtime. As a result, there is currently a great deal of memory that is pre-allocated, but wasted due to the presence of many inactive elements in each underlying subdomain. To mitigate unused memory proliferation, we allocate dense storage for the finest grid only. Adaptively refined finite element scalar functions can then be stored on the finest level using appropriate strides, as shown in Figure 7 (right). Again, a more ideal solution is a completely sparse storage format for the exact number of active elements and ghost layers on any given time step, which would be resized during upscaling and downscaling. Just like dynamic load balancing, the reshuffling and sorting of sparse memory structures would come with its own overhead and a multitude of techniques.

Remark 3.1. *In future work, there are plans to rewrite the code framework to take advantage of both dynamic load balancing and sparse memory allocation techniques, in combination with future studies on adaptive mesh refinement in both space and time.*

4 A Fully-Coupled Solver

The previous EV implementation used a nonlinear block Jacobi solver, whereby the Jacobian matrix on each Newton step assumed a decoupled block diagonal form, where each matrix block corresponds to one subdomain. This was achieved by dropping any matrix entries off the block diagonal corresponding to the interaction between elements of different subdomains. In this form, mass conservation is still achieved at each time step (residual calculations are unchanged in the inexact Newton method), but the number of nonlinear iterations required gets larger as the model becomes more elliptic with less fluid compressibility.

Our new implementation includes an improved fully-coupled Newton solver, which does not artificially force the Jacobian matrix to be in block diagonal form. On the interface Γ, entries corresponding to interface flux terms are constructed a nested loop, which first loops over all elements E_A of a subdomain Ω_k near an interface Γ_{kl}, then loops over all elements E_B of the neighboring subdomain Ω_l that are adjacent to E_B across Γ_{kl}. Denote the pressure DOF p_A in E_A and p_B in E_B. Let $ROW(\cdot)$ be the function that assigns each degree of freedom its own row or column in the linear system for the Newton step. Here, contributions are summed to the Jacobian entry at $(ROW(p_A), ROW(p_B))$ and to the residual vector entry at $ROW(p_A)$. This Jacobain entry is in an unstructured part of the matrix, as opposed to flux entries in the interior of the subdomains, which have a regular stencil. This entry was neglected in the nonlinear block Jacobi method, but is present in the fully-coupled Newton solver. The same residual contributions are calculated in both methods.

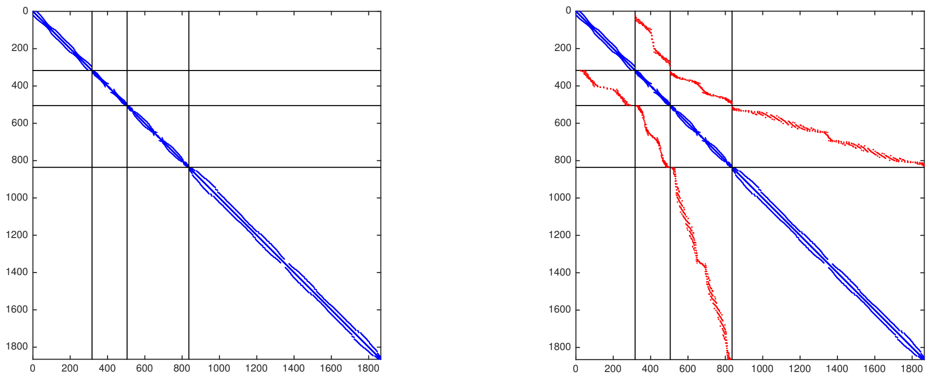


Figure 8: Jacobian matries for the nonlinear block Jacobi solver (left) and the fully-coupled solver (right) for the semi-structured grid in Example 5.1 at the final time step.

In Figure 8, we show the resulting Jacobain matricies for a 4 subdomains case with 8136 elements at a given time step. On the left we show the block diagonal sparsity pattern in the nonlinear block Jacobi method, and on the right we show the sparsity pattern of the fully-coupled Jacobian matrix with entries that couple the subdomains. Blue matrix blocks have a 5-point stencil for $d = 2$, red matrix blocks are unstructured, and lines delineate the subdomain block diagonal structure. In the previous nonlinear block Jacobi solver, only the blue entries were formed, totaling 7740 nonzero entries. In the new fully-coupled Newton solver, both blue and red entries are formed, totaling 9740 nonzero entries.

4.1 Fully-Coupled Solver Performance

Our fully-coupled solver uses the HYPRE library [16] to solve system (8) with an algebraic multigrid (AMG) preconditioned GMRES linear solver. With the single phase slightly compressible flow model, we compare the performance of the previous nonlinear block Jacobi solver with the new fully-coupled Newton solver for the EV method.

To conduct the study, we fix a problem with domain Ω size $25 \times 1000 \times 500$ $[ft^3]$. It is evenly divided into $N = 2$ side-by-side subdomains with uniform $5 \times 10 \times 10$ and $7 \times 13 \times 13$ grids. Initial pressure is $p_0 = 2000$ $[psi]$, and there two vertical wells in a quarter five-spot pattern, with one pressure specified injector at a bottom hole pressure (BHP) of 3000 $[psi]$ and one pressure specified producer at a BHP of 1000 $[psi]$ in opposite corners. Wells are implemented with a Peaceman model [27] to calculate the source function q. Other model parameters include permeability $K = diag(1, 20, 10)$ $[md]$, porosity $\phi = 0.2$, viscosity $\mu = 2$ $[cp]$, fluid density $\rho_{ref} = 56$ $[lb/ft^3]$, time step size $\Delta t = 1$ $[day]$, final time $T = 100$ $[days]$, and no gravity term. Figure 9 shows a typical simulated pressure at the final time.

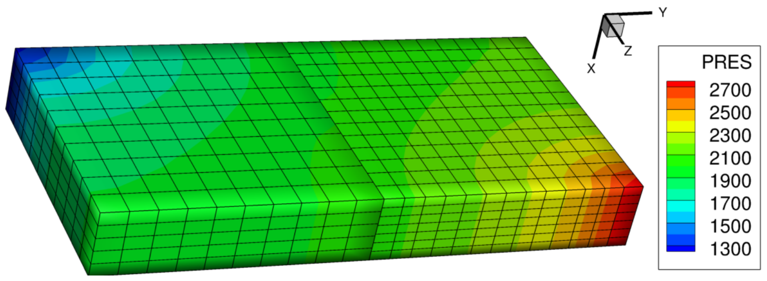


Figure 9: Example for comparing solver performance of nonlinear block Jacobi versus fully-coupled solvers.

Table 1 compares six cases. The first three cases use the nonlinear block Jacobi method and fluid compressibilities $c = \{1\text{E-}4,\ 1\text{E-}5,\ 0\}$ $[1/psi]$. As fluid compressibility decreases $c \to 0$, the second order PDE changes from transient parabolic type to steady-state elliptic type. For this reason, only a single time step is taken for the incompressible case $c = 0$. The last three cases use the fully-coupled solver with the same three compressibilities. We report the average number of linear iterations per newton step, the average number of Newton iterations per time step, the total runtimes for a single processor test, and the speedup of the fully-coupled solver versus the nonlinear block Jacobi solver.

Solver	Compressibility	Lin./Newt.	Newt./Time	Runtime	Speedup
Nonlinear Block	1E-4	9.00	16.46	8.886	
Jacobi	1E-5	11.06	39.98	23.041	
	0	12.00	627.00	3.835	
Fully-Coupled	1E-4	9.00	3.26	1.988	(4.47x)
	1E-5	11.65	2.16	1.526	(15.1x)
	0	12.00	2.00	0.330	(11.62x)

Table 1: Comparison of results with nonlinear block Jacobi and fully-coupled solvers for various fluid compressibilities for the given problem.

A clear trend is observed when using the nonlinear block Jacobi solver. As fluid compressibility decreases, the average number of nonlinear iterations per time step increases greatly. The worst-case scenario for nonlinear block Jacobi is the incompressible case with an average of 627 nonlinear iterations per time step. It performs the worst in this case because the neglected Jacobian matrix entries prohibit the instantaneous global transfer of information necessary to solve a purely elliptic problem. The nonlinear iteration still converges to the correct solution as an inexact Newton method with the correct residual. For fully-compressible problems (e.g. with a gas phase), information propagates more locally and the nonlinear block Jacobi approximation requires fewer inexact Newton iterations.

In the fully-coupled case, the number of nonlinear iterations per time step is very low, comparable to an analogous conforming single domain problem. The best-case scenario for the fully-coupled solver is the incompressible case, where Newton iterations are minimal. The number of linear iterations per Newton step is seen to increase slightly with the fully-coupled solver because it has more nonzero matrix entries and portions of unstructured matrix blocks. This causes the AMG preconditioner to perform slightly worse, but overall the cost of solving one fully-coupled linear system is roughly the same as the block Jacobi system in terms of both runtime and linear iterations. The most important factor influencing runtime is the increased number of nonlinear iterations per time step, which directly translates into significant speedups for the fully-coupled solver over the nonlinear block Jacobi method in these three cases.

5 Numerical Examples

The following four numerical examples will demonstrate the new AMR techniques with the EV method on semi-structured grids. The first three numerical examples will use a multiphase compositional equation of state flow model, and will therefore employ the *a priori* refinement algorithm and corresponding upscaling technique. For specific details on the multiphase compositional flow model, we again refer the reader to [20]. The first example is a homogeneous 2D case, the second example is a highly heterogeneous 2D case, and the third example is a homogeneous 3D case with mesh refinement in three spatial directions. In the fourth numerical example, we switch to the simpler slightly compressible single phase flow model in order to demonstrate the *a posteriori* refinement algorithm using a straightforward residual based error indicator. Our convention is that x-component represents the depth dimension in a right-handed coordinate system. All examples incorporate the effects of gravity, and start from a hydrostatic initial condition.

5.1 Compositional flow, *a priori* refinement, homogeneous 2D case

The first two numerical examples share the same geometry, whereby the domain Ω has aerial size of 1600×1600 $[ft^2]$. The top of the domain resides 8000 $[ft]$ deep and has 5 $[ft]$ thickness. It is considered a 2D problem because each subdomain is 1 element deep. In the aerial plane, there are $N = 4$ refinement levels (i.e. subdomains) consisting of uniform grids ranging from 20×20 on Ω_1 to 160×160 grid on Ω_4. Adaptive time stepping is used which starts at $\Delta t = 0.01$ $[days]$, has a time step multiplier of 1.1, has a maximum time step of $\Delta t = 0.1$ $[days]$, and has a minimum time step of $\Delta t = 1.\text{e-}5$ $[days]$. The final simulation time is $T = 20$ $[days]$. There are no-flow boundary conditions.

The compositional model is configured as a three phase oil-water-gas system (subscripts o, w, g) with several hydrocarbon components. The water phase is slightly compressible. Hydrocarbon components are miscible within fully compressible oil and gas phases, with the phase behavior determined by flash calculations using a cubic Peng-Robinson equation of state. Hence, the model allows mass transfer of hydrocarbon components between oil and gas phases, but the water component only exists in the water phase. The model is isothermal with a reservoir temperature of

160°C. Inputs include known functions for relative permeability and capillary pressure that depend upon phase saturations. A zero diffusion-dispersion tensor is assumed.

The initial conditions are water saturation $s_w(0) = 0.2$, gas saturation $s_g(0) = 0.0$, and oil pressure $p_o = 2000$ [psi]. The water phase properties are compressibility $c_w = 3.3\text{e-}6$ [1/psi], viscosity $\mu_w = 0.7$ [cp], and reference pressure $p_{w,ref} = 0$ [psi]. Standard densities are $\rho_{w,ref} = 62.4$ [psi], $\rho_{o,ref} = 56.0$ [psi], and $\rho_{g,ref} = 0.04228$ [psi]. The composition of the reservoir has 6 components, named C1, C3, C6, C10, C15, and C20. For the sake of brevity, please refer to [20]; Table 3 contains compositional model parameters and binary interaction coefficients, and Figure 3 contains tables for relative permeabilities and capillary pressures. Rock compressibility is $c_r = 5\text{e-}6$ $[1/psi]$. The initial reservoir composition is 60% C6 and 40% C20.

In Example 5.1, the absolute permeability is a scalar $K = 50$ $[md]$ and the porosity is $\phi = 0.3$. Wells are placed in five-spot pattern, with four injectors in the centers of the four quadrants and a producer in the center of the domain. The injection wells are for the water phase at a specified BHP of 4000 $[psi]$ and the production well specifies a BHP of 2000 $[psi]$. These conditions generate a water flood example, where no gas is present throughout the entire simulation. The injected water displaces the resident oil phase, which results in production at the center well. Because of the homogeneous reservoir properties, transient water saturation fronts expand outward from the four injection wells in radial patterns.

To capture a sharp water saturation front in our AMR simulation, the *a priori* error indicator is chosen to be $|\Delta s_w| > 0.05$, whereby an element is marked to belong to the finest refinement level whenever its fine scale saturation compared any of it's neighboring elements' saturations exceeds the given threshold. Surrounding unmarked elements are then graded as described in Figure 5 (Left), and then the given time step is then resolved with the adaptive grid as described in Figure 4 (Left). During this process, upscaling and downscaling of all necessary state variables are performed as described in Figure 6. Snapshots of the corresponding dynamic AMR grids for the 20 $[day]$ compositional EV simulation are presented in Figure 10.

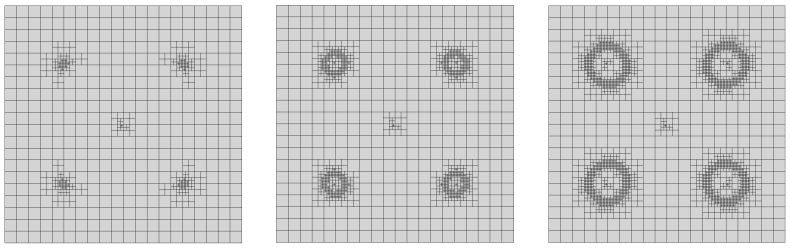


Figure 10: From left to right: AMR grids for Example 5.1 at $t = 0.1$, $t = 3$, and $t = 20$ [days].

To show the quality of the solution, we zoom in on the upper left injection well in Figure 11 to show the final water saturation with the AMR solution versus a solution obtained with a more expensive static fine grid. The choice of indicator allowed the AMR solution to resolve a sharp saturation front in close agreement with the fine scale solution. Note that we enforced elements containing wells to belong to the finest level in the adaptive solution, to minimize recalculations of well parameters, thereby limiting some de-refinements behind the saturation front. The values of the six hydrocarbon component concentrations show similar agreement between AMR and fine scale solutions near the injection wells. Figure 12 shows a comparison of the final oil pressure obtained between the AMR and fine scale solutions. Since the indicator only refined near the saturation front, larger differences are observed in the pressure field where the AMR grid is coarser.

In Figure 13 (Left) we report the number of number of elements used in the AMR solution

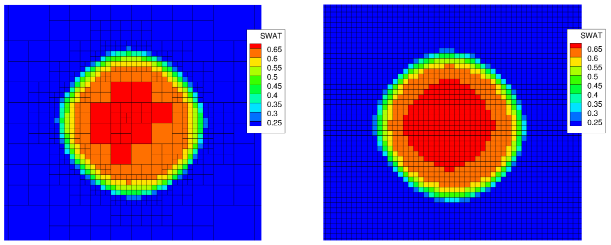


Figure 11: Final water saturation for Example 5.1 near one injection well with the AMR solution (left) versus a fine scale solution (right).

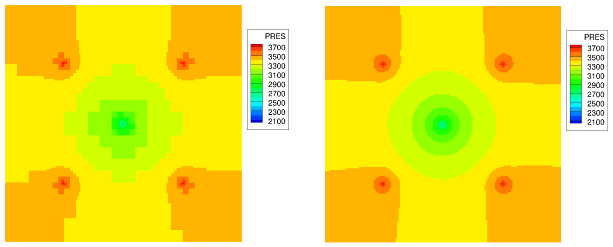


Figure 12: Final oil pressure for Example 5.1 with the AMR solution (left) versus a fine scale solution (right).

versus time as a percentage of the total number of elements used in a fixed fine scale solution grid. For indicator based on gradient of water saturation in this 2D homogeneous case, the AMR solution only required at most 11.38% of the number of elements required in a fine scale simulation. Due to the fewer number of unknowns, this had a direct impact on the runtime required for the linear solver. For a single processor simulation, the total runtime spent solving linear systems was 90.17 $[sec]$ for the fine scale problem, and was reduced to 9.61 $[sec]$ for linear solver runtime on the AMR grid, a speedup factor of 9.38x. In Figure 13 (Right), we compare an output quantity of interest between the AMR and fine scale solutions, namely the cumulative oil production in the center well. This homogeneous 2D case showed a very good match, differing by only 0.47% at $t = 20$ $[days]$.

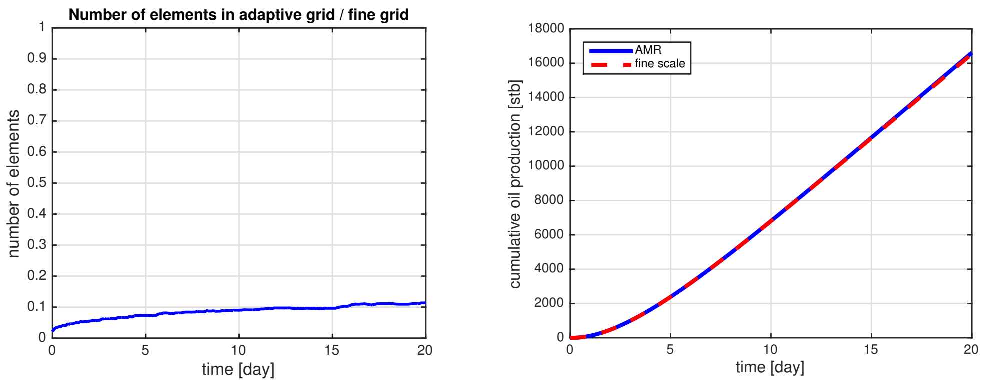


Figure 13: In Example 5.1, number of elements used in the AMR solution (left), and comparison of cumulative oil production (right).

5.2 Compositional flow, *a priori* refinement, heterogeneous 2D case

Example 5.2 has the same 2D geometry and same compositional model parameters as described in Example 5.1. The differences in this example are that the permeability and porosity fields are highly heterogeneous, and that pairs of injection and production wells are placed at opposite ends of three high permeability channels. The fine scale heterogeneous permeability and porosity data was assembled on the 160×160 fine grid by concatenating properties from layers 37, 38, and 39 of the SPE10 dataset [13] as shown on a logarithmic scale in Figure 14. Harmonic upscaling was used to generate the coarser permeability levels, and arithmetic upscaling was used to generate the coarser porosity levels. Injection and production well locations are indicated on the finest level with blue and brown arrows, respectively. The initial conditions and BHP specified well conditions also match Example 5.1. Adaptive time stepping parameters are kept the same, but the final simulation time is $T = 80$ $[days]$.

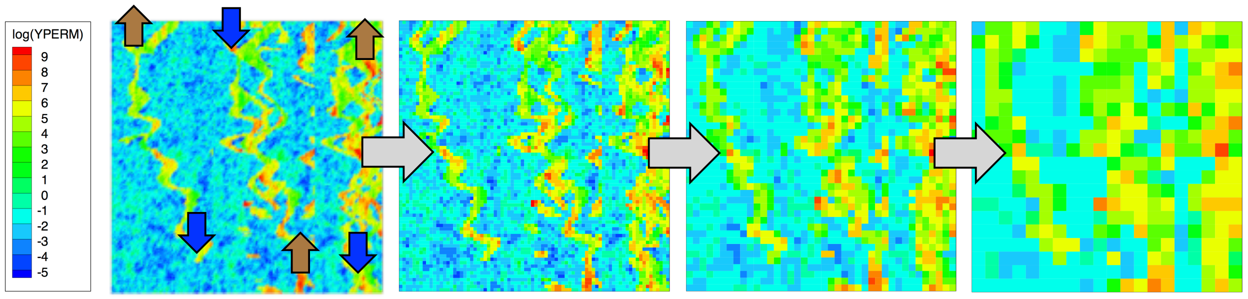


Figure 14: In Example 5.2, fine scale heterogeneous permeability was upscaled to generate properties on the three coarser levels.

Two AMR simulations are performed in Example 5.2 using two different *a priori* error indicators:

- Indicator #1 is the condition $|\Delta s_w| > 0.01$.
- Indicator #2 is are the conditions $|\Delta s_w| > 0.01$ or $|\Delta p_o| > 10$ $[psi]$.

Snapshots of the corresponding dynamic AMR grids with both indicators are presented in Figure 15. The reason we have chosen to use a second indicator is due to the extreme heterogeneity. The evolving velocity fields of each fluid phase are important to advect saturations and concentrations within preferential pathways, as transient fronts are not expected to expand from injection sources in radial patterns. Since the first indicator is solely focused on resolving the sharp saturation front, it fails to sharply resolve velocity fields away from the front, which paradoxically has a negative effect on predicting where that front should evolve. Conversely, the second indicator marks many more cells because it also tries to better resolve locations with sharp gradients in oil phase pressure. This better predicts phase velocity fields for advection, but it is more costly as there are clearly more areas of refinement. Figure 16 shows a comparison of oil pressure fields at an early time of $t = 10$ $[days]$. The AMR solution with Indicator #1 has a much less accurate pressure field away from the saturation front, while Indicator #2 more closely matches the fine scale pressure. Figure 17 shows plots of the final water saturation field. As a direct consequence of having better velocity fields on which to advect the saturation fronts, the AMR solution #2 more closely matches the fine scale saturation field compared to AMR solution #1. The bottom row of this figure shows a closeup view of the grid around the production well in the upper-right corner of the domain. AMR solution #1 over-predicted how far the front has traveled, while AMR solution #2 more closely matched the front to the fine scale solution.

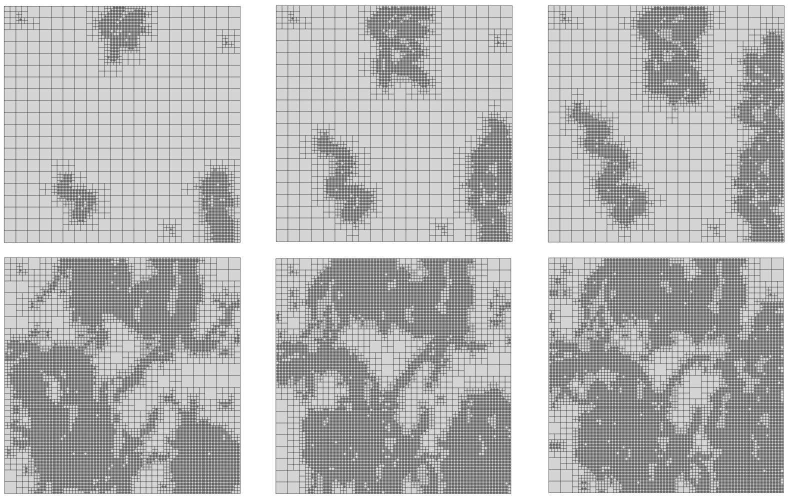


Figure 15: From left to right: AMR grids for Example 5.2 at $t = 10$, $t = 40$, and $t = 80$ [days]. The top row shows the AMR solution with Indicator #1 and the bottom row shows the AMR solution with Indicator #2.

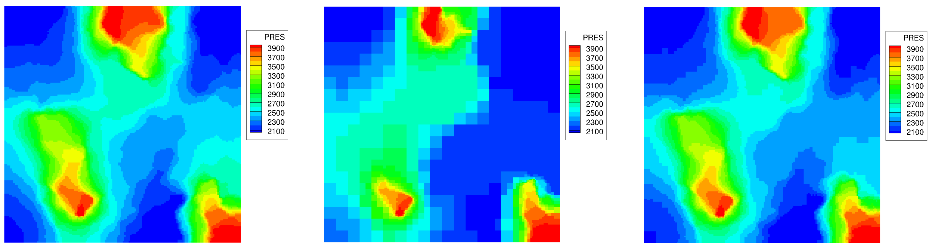


Figure 16: From left to right: comparison of oil pressure fields at $t = 10$ [$days$] for fine scale, AMR with Indicator #1, and AMR with Indicator #2 solutions.

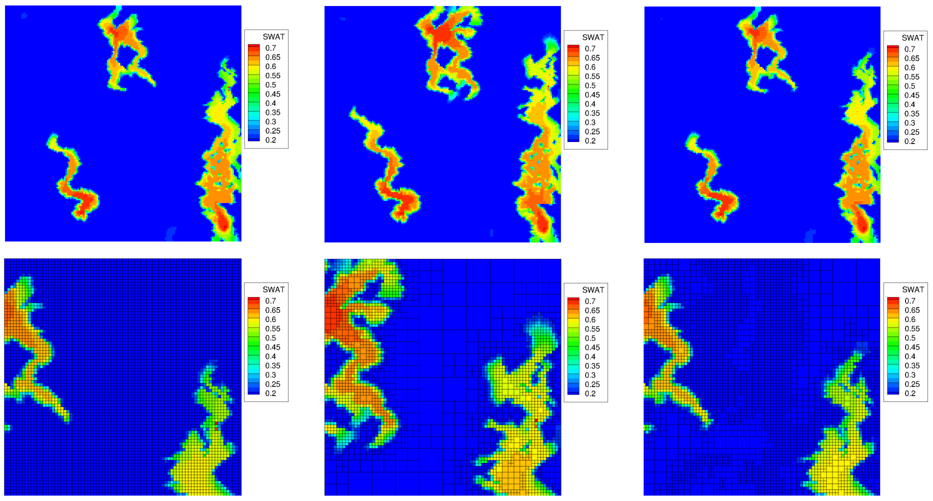


Figure 17: From left to right: comparison of water saturation at $t = 80$ [$days$] for fine scale, AMR with Indicator #1, and AMR with Indicator #2 solutions. Top row shows the entire domain, and bottom row shows the grid near the upper right production well.

In Figure 18 (Left) we report the number of number of elements used in both AMR solutions versus time as a percentage of the total number of elements used in a fixed fine scale solution grid. AMR solution #1 used at most 30.21% of the number of fine scale elements, while AMR solution #2 consistently used closer to the maximum of 69.48% of the number of fine scale elements. We ran this simulation with 4 processors. For linear solver runtime, the fine grid took 159.91 [sec], AMR solution #1 took 108.64 [sec], and AMR solution #2 took 272.70 [sec]. This meant AMR solution #1 was 1.47x faster than the fine scale, while AMR solution #2 was 0.59x slower than the fine scale. It is possible that better solver parameters could improve these results, or that for this fully-compressible model the nonlinear block Jacobi solver might outperform the fully-coupled solver. In Figure 18 (Right), we compare an output quantity of interest between the two AMR solutions versus the fine scale solution, namely the cumulative oil production in the upper right production well. For the heterogeneous 2D case, AMR solution #1 had an error of 43.51% and AMR solution #2 had an improved error of 21.96% at $t = 80$ [$days$], but with additional computational cost.

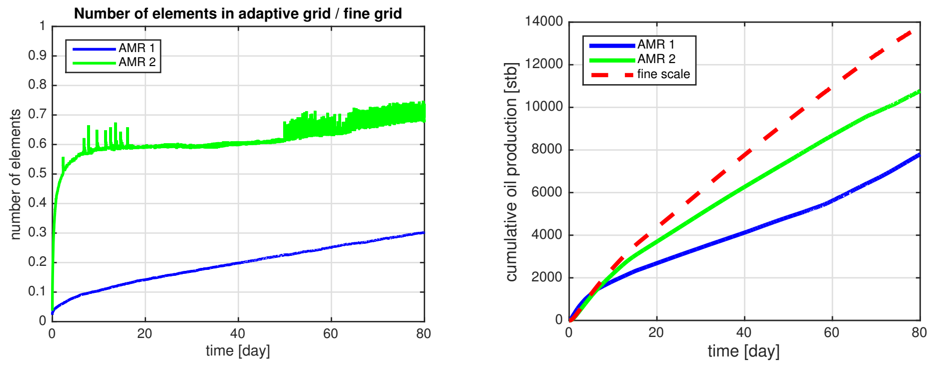


Figure 18: In Example 5.2, number of elements used in the two AMR solutions (left), and comparison of cumulative oil production at the upper right well(right).

5.3 Compositional flow, *a priori* refinement, homogeneous 3D case

Example 5.3 again has homogeneous reservoir properties, but demonstrates that local refinements are possible in the EV method in three dimensions. The reservoir domain Ω has size $20 \times 1600 \times 1600$ [ft^3] at a top depth of 8000 [ft] deep. In this case there are $N = 4$ refinement levels consisting of uniform grids ranging from $2 \times 20 \times 20$ on Ω_1 to $16 \times 160 \times 160$ grid on Ω_4. Time step parameters follow the preceding two examples, and the final simulation time is $T = 20$ [$days$]. There same five-spot well pattern is used as Example 5.1, with vertical wells completed through the entire reservoir depth at the same specified BHP.

The *a priori* error indicator used is $|\Delta s_w| > 0.5$ to resolve a sharp water saturation front. Figure 19 shows the AMR solution at several time steps with transparent grids on the left half of the domain with opaque water saturation iso-surfaces shown. As the saturation front sweeps out from the injection wells, the grids are adaptively refined in all three dimensions. The bottom row of the figure shows a vertical cross-sectional view, making it clear that refinements are also happening in the depth dimension. Note there is slightly more refinement at the bottom of the domain compared to the top, because of gravitational effects on the fluid density.

Figure 20 shows the maximum number of elements used in the 3D AMR solution was at most 8.71% of the number of fine scale elements, and that the error in cumulative oil production was 1.20% for the 3D AMR solution versus the fine scale solution at $t = 20$ [$days$]. The simulation

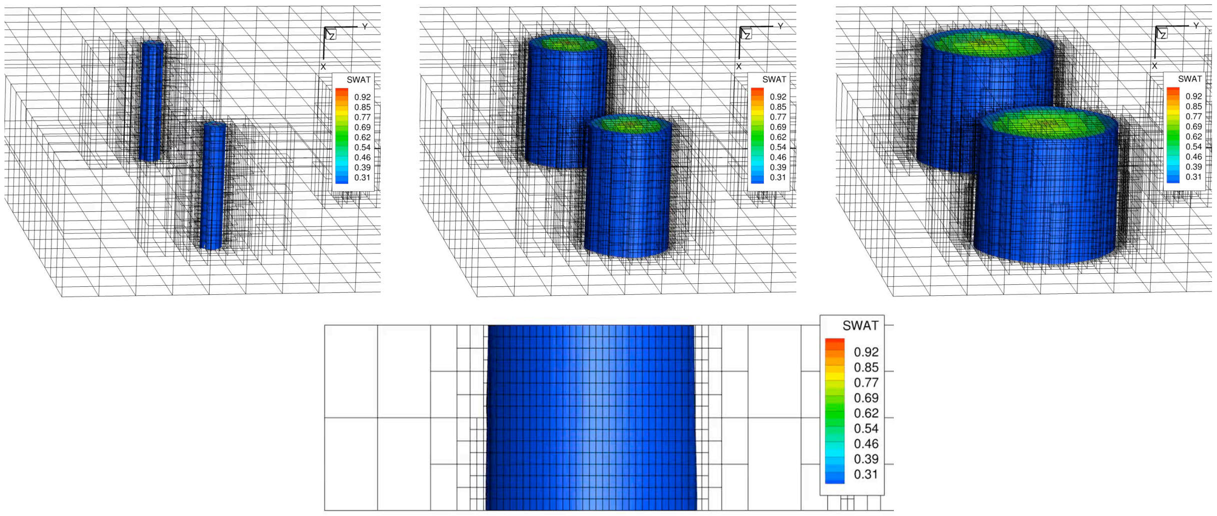


Figure 19: From left to right: transparent AMR grids with water saturation iso-surfaces for near the two injection wells in the left half of the domain for Example 5.3 at $t = 0.1$, $t = 3$, and $t = 20$ $[days]$. Bottom row shows a vertical cross-sectional view at $t = 20$ $[days]$.

was run with on single processor. The linear solver had a runtime of 1536.3 $[sec]$ for the fine scale problem versus a runtime of 85.6 $[sec]$ for the AMR problem, giving a speedup factor of 18x. As expected, the computational efficiency of AMR improved 3D versus 2D, albeit for a homogeneous problem.

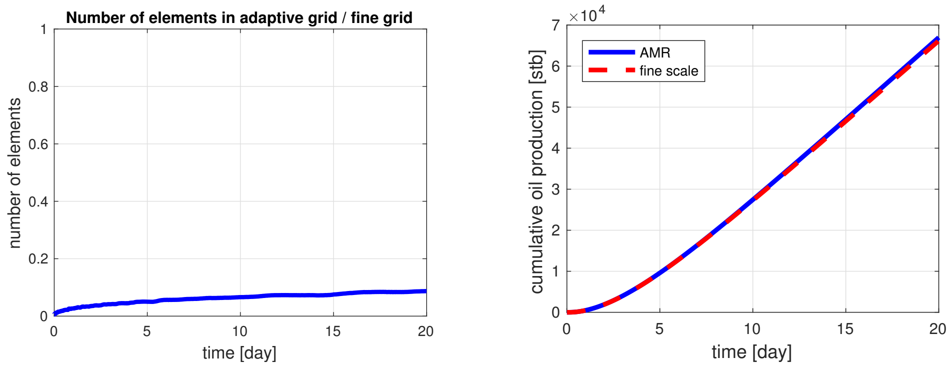


Figure 20: In Example 5.3, number of elements used in the AMR solution (left), and comparison of cumulative oil production (right).

5.4 Single phase flow, *a posteriori* refinement, homogeneous 2D case

In the final example, we apply a simple residual-based *a posteriori* error indicator to a slightly compressible single phase flow problem. The 2D domain, discretization, permeability, porosity, and well pattern match Example 5.1. The single phase model parameters are reference density $\rho_{ref} = 56$ $[lb/ft^3]$, fluid viscosity $\mu = 2$ $[cp]$, and fluid compressibility $c = 1.\text{e-}5$ $[1/psi]$. In this example we have a uniform time step $\Delta t = 0.01$ $[days]$ and a final simulation time $T = 20$ $[days]$.

The *a posteriori* error indicator used is related to the normalized nonlinear residual of the weak form. We note that a Newton method can make the nonlinear residual arbitrarily small on any given grid, but that a "good" error will be distributed evenly throughout the domain, an observation also noted by others [6]. The basic idea is that the grid will be adapted whenever there are spikes in the normalized nonlinear residual. Other possible error indicators are ones based upon the nonlinear residual of the strong form, or a quantity related to the curvature of the solution. We reiterate that the goal of this paper is not in the development of new error indicators, but just to demonstrate that our new AMR procedure can successfully be driven by *a posteriori* procedures. Unlike the previous three examples, the elements are successively marked for refinement using the algorithm given in Figure 5 (Right), the refinement algorithm used is described in Figure 4 (Right). The downscaling procedure used is described in Figure 6, and no de-refinement (i.e. upscaling) is performed in this example.

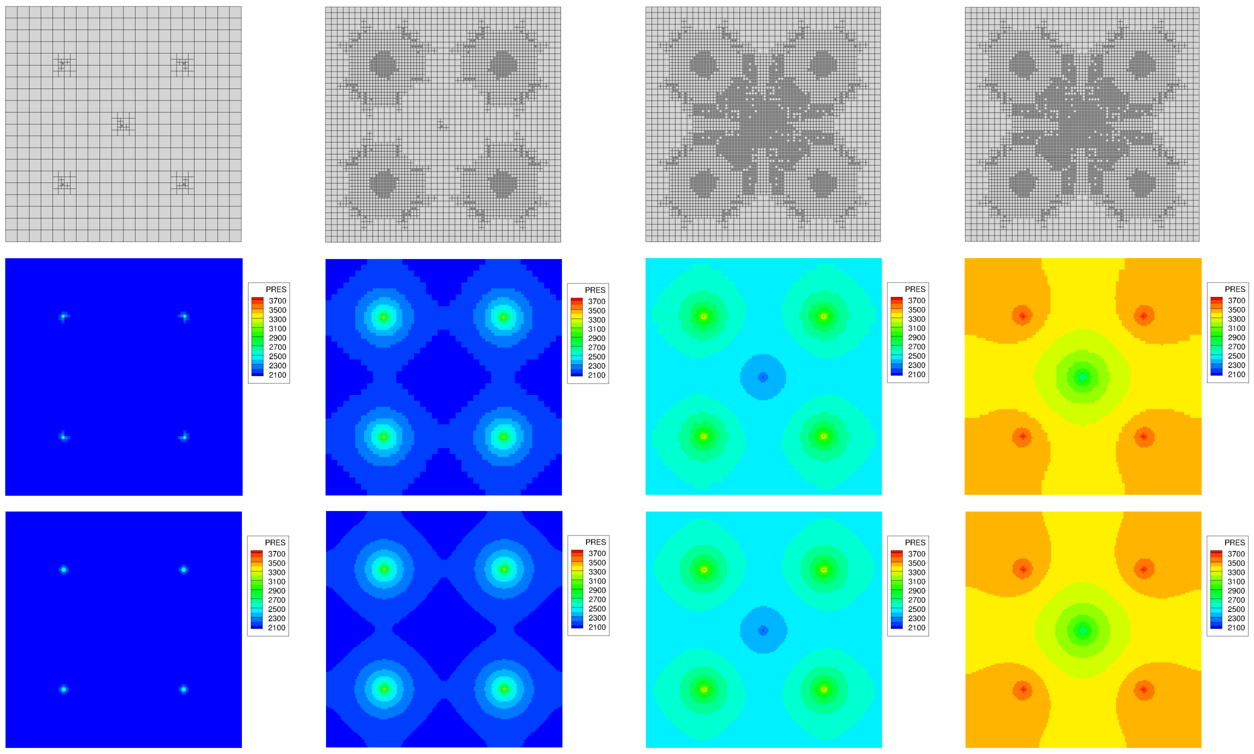
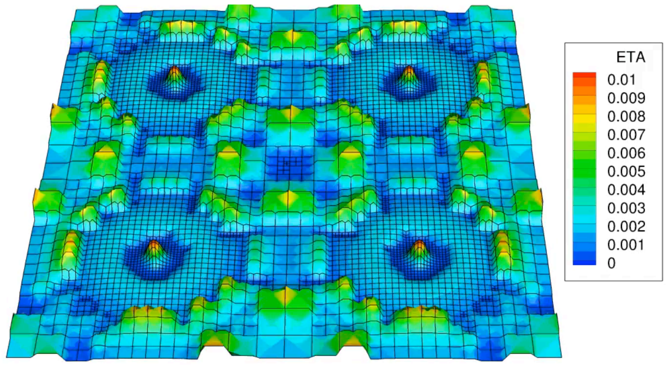


Figure 21: From left to right: AMR grids for Example 5.4 at $t = 0.1$, $t = 1$, $t = 4$, and $t = 20$ $[days]$. Top row shows AMR grid, middle row shows AMR solution pressure, bottom row shows fine scale solution pressure.

The numerical results are summarized as follows. From left to right, the four columns of Figure 21 present the numerical solution at simulation times $t = 0.1$, $t = 1$, $t = 4$, and $t = 20$ $[days]$. The first row of this figure shows the AMR grids, and how they were adapted to the *a posteriori* indicator based upon the normalized nonlinear residual at each time step. The second row shows the resulting pressure solution in the five-spot well test. The third row shows the results of a fine scale simulation that was run with a static uniform 160×160 grid. As the plots show, the AMR solution is in close agreement with the fine scale solution. A snapshot of the *a posteriori* indicator is shown in Figure 22 (left), with the height dimension the value of the indicator. Whenever the peaks of this surface exceed a threshold, those grid elements are refined by one level for the subsequent time step. In Figure 22 (right), we report the number of elements used in the adaptive solution as a percentage of the total number of elements in the fine scale. Since there was no de-refinement in this scheme, the curve is monotone increasing, and has a maximum value of 32.75%. The AMR and fine scale simulations were run on a single processor. Linear solver runtime decreased from 138.76 [*sec*] in the fine scale example to 50.52 [*sec*] in the AMR solution. Since the fine scale solution is completely decoupled from the *a posteriori* AMR solution, we report that setup time for the linear systems decreased from 40.32 [*sec*] in the fine scale to 16.31 [*sec*] in the AMR solution.



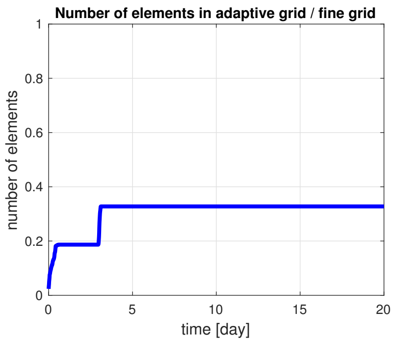


Figure 22: In Example 5.4, a snapshot of the normalized nonlinear residual-based *a posteriori* error at $t = 1$ [*days*] (left), and the number of elements used in the AMR solution (right).

6 Conclusions

In this work, we have formulated the enhanced velocity mixed finite element method on a new type of semi-structured grid for multiple models describing both single and multiphase flow in porous media. We have also generalized the method to support dynamic adaptive mesh refinement, and applied refinement and upscaling algorithms using *a priori* and *a posteriori* based error indicators. We have also implemented a fully-coupled solver that forms unstructured matrix blocks corresponding to the direct interactions of elements belonging to different subdomains. We have demonstrated the fully-coupled solver requires fewer nonlinear iterations and computational runtime versus the nonlinear block Jacobi method, especially for flow models with a strong elliptic character. We have also demonstrated that posing the EV method on semi-structured grids gives an easy way to enable powerful local refinement that can accurately and efficiently capture transient flow features. Ongoing research is combining these adaptive mesh refinement techniques in concert with fracture propagation and geomechanics, with space-time methods to allow local time stepping, and in the development of better *a posteriori* error indicators.

Acknowledgements

This work was supported by DOE grant FG02-04ER25617, NSF grant 1546553, the CSM industrial affiliates program, and the Center for Frontiers of Subsurface Energy Security (CFSES), an Energy Frontier Research Center funded by the U.S. Department of Energy, Office of Science, Office of Basic Energy Sciences under Award Number DE-SC0001114.

References

[1] I. Aavatsmark. An introduction to multipoint flux approximations for quadrilateral grids. *Computational Geosciences*, 6(3-4):405–432, 2002.

[2] M. Ainsworth and J. T. Oden. *A Posteriori Error Estimation in Finite Element Analysis*, volume 37. John Wiley & Sons, 2011.

[3] Y. Amanbek, G. Singh, M. F. Wheeler, and H. van Duijn. Adaptive numerical homogenization for upscaling single phase flow and transport. Technical report, The University of Texas at Austin, 2017. ICES Report 17-12.

[4] T. Arbogast, G. Pencheva, M. F. Wheeler, and I. Yotov. A multiscale mortar mixed finite element method. *Multiscale Modeling & Simulation*, 6(1):319–346, 2007.

[5] T. Arbogast, M. F. Wheeler, and I. Yotov. Mixed finite elements for elliptic problems with tensor coefficients as cell-centered finite differences. *SIAM Journal on Numerical Analysis*, pages 828–852, 1997.

[6] W. Bangerth and R. Rannacher. *Adaptive Finite Element Methods for Differential Equations.* Birkhäuser, 2013.

[7] M. Bause and P. Knabner. Computation of variably saturated subsurface flow by adaptive mixed hybrid finite element methods. *Advances in Water Resources*, 27(6):565–581, 2004.

[8] F. Brezzi, J. Douglas, R. Durán, and M. Fortin. Mixed finite elements for second order elliptic problems in three variables. *Numerische Mathematik*, 51(2):237–250, 1987.

[9] F. Brezzi, J. Douglas, and L. D. Marini. Two families of mixed finite elements for second order elliptic problems. *Numerische Mathematik*, 47(2):217–235, 1985.

[10] F. Brezzi and M. Fortin. *Mixed and Hybrid Finite Element Methods*, volume 15. Springer Science & Business Media, 2012.

[11] C. Burstedde, L. Wilcox, and O. Ghattas. p4est: Scalable algorithms for parallel adaptive mesh refinement on forests of octrees. *SIAM Journal on Scientific Computing*, 33(3):1103–1133, 2011.

[12] C. Carstensen and R. Hoppe. Error reduction and convergence for an adaptive mixed finite element method. *Mathematics of Computation*, 75(255):1033–1042, 2006.

[13] M. A. Christie and M. J. Blunt. Tenth SPE comparative solution project: a comparison of upscaling techniques. *SPE Reservoir Evaluation & Engineering*, 4(4):308–317, 2001.

[14] M. Delshad, X. Kong, R. Tavakoli, S.A. Hosseini, and M. F. Wheeler. Modeling and simulation of carbon sequestration at cranfield incorporating new physical models. *International Journal of Greenhouse Gas Control*, 18:463–473, 2013.

[15] F. Drui, A. Fikl, P. Kestener, S. Kokh, A. Larat, V. Le Chenadec, and M. Massot. Experimenting with the p4est library for amr simulations of two-phase flows. *ESAIM: Proceedings and Surveys*, 53:232–247, 2016.

[16] R. Falgout and U. Yang. HYPRE: A library of high performance preconditioners. *Computational Science, ICCS 2002*, pages 632–641, 2002.

[17] R. Finkel and J. Bentley. Quad trees a data structure for retrieval on composite keys. *Acta Informatica*, 4(1):1–9, 1974.

[18] B. Ganis, M. Juntunen, G. Pencheva, M. F. Wheeler, and I. Yotov. A global jacobian method for mortar discretizations of nonlinear porous media flows. *SIAM Journal on Scientific Computing*, 36(2):A522–A542, 2014.

[19] B. Ganis, K. Kumar, G. Pencheva, M. F. Wheeler, and I. Yotov. A global jacobian method for mortar discretizations of a fully implicit two-phase flow model. *Multiscale Modeling & Simulation*, 12(4):1401–1423, 2014.

[20] B. Ganis, G. Singh, and M. F. Wheeler. A parallel framework for a multipoint flux mixed finite element equation of state compositional flow simulator. *Computational Geosciences*, 21(5-6):1189–1202, 2017.

[21] B. Ganis, M. F. Wheeler, and I. Yotov. An enhanced velocity multipoint flux mixed finite element method for darcy flow on non-matching hexahedral grids. *Procedia Computer Science*, 51:1198–1207, 2015.

[22] B. Ganis and I. Yotov. Implementation of a Mortar Mixed Finite Element Method using a Multiscale Flux Basis. *Comput. Methods Appl. Mech. Engrg.*, 198(49-52):3989–3998, 2009.

[23] R. Glowinski and M. F. Wheeler. Domain decomposition and mixed finite element methods for elliptic problems. In *First International Symposium on Domain Decomposition Methods for Partial Differential Equations*, pages 144–172, 1988.

[24] R. Hoppe and B. Wohlmuth. Adaptive mixed hybrid and macro-hybrid finite element methods. *Acta Math. Univ. Comenianae*, 67(1):159–179, 1998.

[25] K. Lipnikov, M. Shashkov, and I. Yotov. Local flux mimetic finite difference methods. *Numerische Mathematik*, 112(1):115–152, 2009.

[26] J.-C. Nédélec. Mixed finite elements in $\mathbb{R}^3$. *Numerische Mathematik*, 35(3):315–341, 1980.

[27] D.W. Peaceman. Interpretation of well-block pressures in numerical reservoir simulation with nonsquare grid blocks and anisotropic permeability. *SPE Journal*, 23(3):531–543, 1983.

[28] G. Pencheva, M. Vohralik, M. F. Wheeler, and T. Wildey. Robust a posteriori error control and adaptivity for multiscale, multinumerics, and mortar coupling. *SIAM Journal on Numerical Analysis*, 51(1):526–554, 2013.

[29] G. Pencheva and I. Yotov. Balancing domain decomposition for mortar mixed finite element methods. *Numer. Linear Algebra Appl*, 10:159–180, 2003.

[30] P. Raviart and J. Thomas. A mixed finite element method for 2nd order elliptic problems. *Mathematical Aspects of Finite Element Methods*, pages 292–315, 1977.

[31] T. Russell and M. F. Wheeler. Finite element and finite difference methods for continuous flows in porous media. In *The Mathematics of Reservoir Simulation*, pages 35–106. SIAM, 1983.

[32] K. Schloegel, G. Karypis, and V. Kumar. Multilevel diffusion schemes for repartitioning of adaptive meshes. *Journal of Parallel and Distributed Computing*, 47(2):109–124, 1997.

[33] G. Singh, Y. Amanbek, and M. F. Wheeler. Adaptive numerical homogenization for non-linear multiphase flow and transport. Technical report, The University of Texas at Austin, 2017. ICES Report 17-13.

[34] S. Thomas and M. F. Wheeler. Enhanced velocity mixed finite element methods for modeling coupled flow and transport on non-matching multiblock grids. *Computational Geosciences*, 15(4):605–625, 2011.

[35] J. Wheeler, M. F. Wheeler, and I. Yotov. Enhanced velocity mixed finite element methods for flow in multiblock domains. *Computational Geosciences*, 6(3):315–332, 2002.

[36] M. F. Wheeler, G. Xue, and I. Yotov. A multipoint flux mixed finite element method on distorted quadrilaterals and hexahedra. *Numerische Mathematik*, 121(1):165–204, 2012.

[37] M. F. Wheeler and I. Yotov. A posteriori error estimates for the mortar mixed finite element method. *SIAM Journal on Numerical Analysis*, 43(3):1021–1042, 2005.