FLOW-BASED COARSENING FOR MULTISCALE SIMULATION OF TRANSPORT IN POROUS MEDIA

VERA LOUISE HAUGE, KNUT–ANDREAS LIE, AND JOSTEIN R. NATVIG

Abstract. Geological models are becoming increasingly large and detailed to account for heterogeneous structures on different spatial scales. To obtain simulation models that are computationally tractable, it is common to remove spatial detail from the geological description by upscaling. Pressure and transport equations are different in nature and generally require different strategies for optimal upgridding. To optimize the accuracy of a transport calculation, the coarsened grid should generally be constructed based on *a posteriori error* estimates and adapt to the flow patterns predicted by the pressure equation. However, sharp and rigorous estimates are generally hard to obtain, and herein we therefore consider various ad-hoc methods for generating flow-adapted grids. Common for all, is that they start by solving a single-phase flow problem once and then continue to form a coarsened grid by amalgamating cells from an underlying fine-scale grid. We present several variations of the original method. First, we discuss how to include *a priori* information in the coarsening process, e.g., to adapt to special geological features or to obtain less irregular grids in regions where flow-adaption is not crucial. Second, we discuss the use of bi-directional versus net fluxes over the coarse blocks, and show how the latter gives systems that better represent the causality in the flow equations, which can be exploited to develop very efficient nonlinear solvers. Finally, we demonstrate how to improve simulation accuracy by dynamically adding local resolution near strong saturation fronts.

1. Introduction

The main purpose of reservoir simulation is to provide predictions of the movement of hydrocarbon phases and water that will help oil and gas companies make better decisions on how to develop and produce their assets. The complexity of the workflows that lead to decisions is ever increasing, and advances in reservoir characterization, production optimization, and real-time reservoir management is leading to continued demand for faster and more advanced flow simulation tools. In particular, optimizing the recovery from mature and brown field assets will require multi-fidelity simulators that have a lot of flexibility and scalability to enable reservoir engineers to evaluate many (different) scenarios.

Coming up with a satisfactory solution is a challenging and daunting task and in this paper, we will only address a small part of the problem: development of multi-fidelity transport solvers to overcome the gap in resolution between geological and simulation models. Geo-cellular models resulting from structural and petrophysical modelling typically contain significantly more detail than what the reservoir engineer can afford if the simulation is to fit in memory and finish within a reasonable time frame of what he/she feels is necessary to capture the flow dynamics of a particular scenario with sufficient detail.

The traditional approach has been to use upgridding to create a new grid model with reduced spatial resolution and upscaling to bring petrophysical properties from the high-resolution geological description down to the new grid. A large number of different strategies have been developed to minimize the errors introduced in this model-reduction process, see e.g., [4, 6, 7, 10]. Upgridding and upscaling is generally a manual process and choosing the 'right' method and model resolution can be highly problem dependent and very time-consuming. The problem becomes more complicated as changes are introduced in the reservoir description to match observed (dynamical) data. Ideally, all changes should be made to the fine-scale geological model. However, because the turnaround time of traditional up- and downscaling processes is typically much larger than the man-hours allocated to the modelling project, one ends up with incompatible models at different spatial resolutions.

Herein, we will consider a multiscale approach to geological modelling. This approach differs from the traditional upgridding/upscaling approach in the sense that a fine-grid model is present at all times. Then it is up to the multiscale simulator to (automatically) coarsen the grid to reduce the number of degrees of freedom to a level that is sufficient to resolve flow physics and satisfy requirements on computational costs. The first component in such a simulator is a multiscale flow solver [9, 12, 13] that captures the fluid flow as a linear combination of a set of numerically computed basis functions. As such, the multiscale flow solver can be considered either as a robust upscaling method, or as a single-step upscaling-downscaling method that delivers approximate fine-scale fluxes. The basis functions are computed by solving localized flow problems, and the main distinction between different multiscale methods is how the local flow problems are constructed. The methods presented in the following are developed to accompany a particular method [3], but all ideas presented can readily be combined with any multiscale flow solver that produces *conservative* fluxes on coarse, fine, and intermediate grids. Previous research [3] has shown that our particular multiscale flow solver gives the best performance when the associated coarse grid follows geological structures. For corner-point grids, this is typically achieved through a regular partitioning in index space (using the underlying logical ijk numbering). Figure 1 gives a visual illustration of a multiscale flow solver.

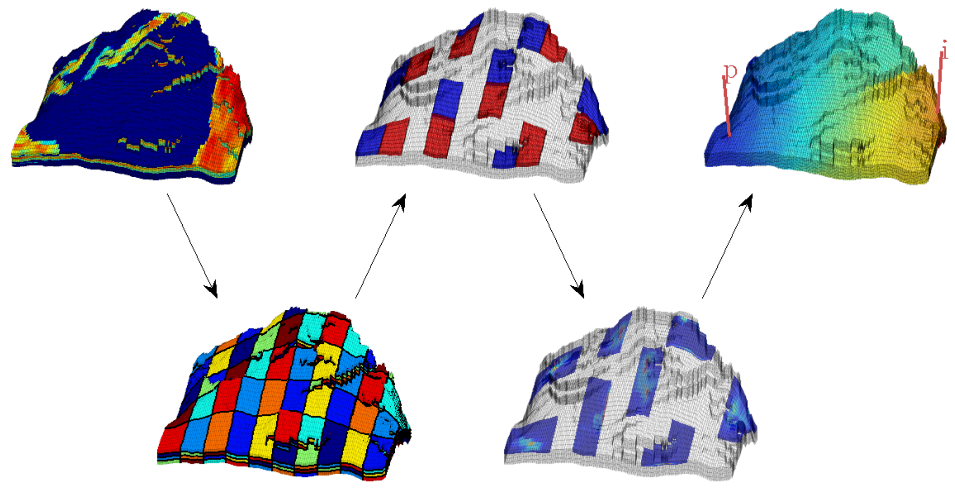


FIGURE 1. Illustration of a multiscale flow solver applied to a model from the SAIGUP study [18]. The coarse grid is generated as a regular partitioning of the underlying fine grid. Localized flow problems are set up (here, between each pair of blocks that share a common face) and then solved numerically. Finally, the basis functions are used to construct a global flow solution.

The second component of a multiscale simulator is an efficient transport solver that uses the flow field (pressure and total velocity) from the multiscale flow solver to evolve fluid saturations and compositions. To this end, there are several alternatives, depending upon which resolution one chooses to use. If the transport is to be computed on the same coarse grid as used in the multiscale flow solver, the best choice is to use a standard implicit finite-volume method with the coarse-scale fluxes computed by the multiscale solver. To solve the transport without upscaling—using the fine-scale, approximate fluxes—the best choice is probably a streamline method, e.g., as described in [1, 21], or one can use similar operator-splitting techniques to develop highly-efficient finite-volume solvers [19, 20] that use flow information to obtain an optimal ordering of the nonlinear discrete transport equations so that these can be solved in a cell-by-cell or block-by-block fashion. This gives local control over the (computationally expensive) nonlinear iterations and can significantly reduce the computational cost compared with standard (implicit) finite-volume methods.

In many cases solving saturation equations on the coarse scale may be too inaccurate and solving it on the fine grid may be too costly, and one would therefore look for a compromise between accuracy and computational speed. The adaptive multiscale finite-volume method of Lee et al. [17] and Zhou et al. [22] is one approach in this direction, in which three prolongation operators with different computational complexity were used to construct a multiscale transport solver. Alternatively, to optimize the accuracy of a transport calculation, the coarsened grid should generally adapt to the flow patterns predicted by the flow solver. Ideally, the flow-adapted grid should be constructed using a posteriori error estimates. Obtaining sharp and rigorous estimates is generally hard and good ad-hoc methods have been shown to capture flow and transport in highly heterogeneous reservoirs with good accuracy [2, 8]. In a flow-based method the grid is aligned to capture high-flow regions and (clearly) distinguish between regions of high and low flow. By capturing these important characteristics of the flow, one is able to generate coarse grids with a high upscaling factor that still deliver good accuracy.

In this paper, we will combine several of these ideas for speeding up the transport solves in a multiscale simulator setting. First of all, we introduce important algorithmic improvements for the nonuniform coarsening method of Aarnes et al. [2]: In our experience, the original algorithm will in many cases introduce too much flow adaption and create grids that are unnecessary irregular. To countermand this and to increase the flexibility of the transport grids, we extend the nonuniform coarsening algorithm to include *a priori* grid partitions that can be combined with (local) flow adaption. The *a priori* partitions can be constructed from the geometry/topology of the grid (e.g., as a regular or tensorial partition in index space of a logically Cartesian grid or as a standard graph-partition [14] of an unstructured grid), be based on features in the underlying geology (e.g., following features in the stratigraphic architecture or adapting to saturation regions to reduce the need for multiphase upscaling), or simply impose the user's perception of a good grid based on experience or expert knowledge. Moreover, to improve the regularity of the flow-adapted coarse blocks, we use a different neighbour definition when amalgamating cells from the fine grid into coarse blocks. Finally, we use an alternative flux approximation for the coarse-scale solver to reduce the numerical dissipation and reduce the couplings in the corresponding nonlinear discrete system.

Altogether, the resulting algorithm can be used to generate static flow-based grids that give reasonable compromises between efficiency and accuracy. However, in cases where the dynamics of the problem is dictated by strong (saturation or component) fronts, it may not be possible to create a good *static* grid. Inspired by the adaptive multiscale method of Lee et al. [17] and Zhou et al. [22], we therefore also demonstrate how one can develop nonuniform grids with both static and dynamic flow adaption.

2. Mathematical Model and Numerical Discretization

In general, multi-fidelity simulators need to cover a wide range of enhanced recovery processes including biological, chemical, electrical, gas-based, thermal, and water-based methods. However, in the following we will only consider the simplified setting of an incompressible, immiscible two-phase flow, described by a set of flow equations for the global pressure p and the total velocity $\vec{v}$

$$\nabla \cdot \vec{v} = h_p, \qquad \vec{v} = -\mathbf{K}\lambda(S)\nabla p, \tag{1}$$

and a transport equation for the saturation of one of the phases,

$$\phi \frac{\partial S}{\partial t} + \vec{v} \cdot \nabla f(S) = h_S. \tag{2}$$

These equations are defined over a singly-connected domain, represented by a grid that consists of a set of grid cells c_i, $i = 1, \ldots, n$. No further assumptions are made on the geometry and topology of the grid, apart from the requirement of an explicit mapping $\mathcal{N}(c)$ between cell c and its nearest neighbours. Most of the ideas we present in the following will therefore be applicable to any matching, unstructured, polyhedral grid. To keep the presentation as simple as possible, our examples will, with two exceptions, focus on 2D Cartesian grids, taken from individual layers of the widely used SPE10 benchmark [5]. Likewise, as our interest is primarily in the transport solver, we make no assumptions about the flow solver except that it produces mass-conservative fluxes on each cell. In a multiscale setting, our primary example of such a solver would be the multiscale mixed-finite method of Aarnes et al. [3]. In the following, v_{ij} will denote the flux over fine-cell interface γ_{ij} between cells c_i and c_j.

The remains of the paper will focus on transport solvers defined over a coarse grid that is constructed by grouping sets of cells into blocks B_ℓ, $\ell = 1, \dots, N$. The simplest way of representing such a coarse grid is by a partition vector p with n elements, for which element p_i assumes the value ℓ if cell c_i is member of block B_ℓ. Representing the coarse grid by a partition vector gives us great flexibility in the shape of individual grid blocks and also opens up for a simple (interactive) manual editing, if deemed necessary. Herein, however, we will *not* use any manual editing to improve grid quality. More details of the grid generation will be given from the next section and onward.

Having created a coarse grid, the next step is to construct a coarse-grid transport solver. To this end, we assume that the saturations are constant over each grid block, i.e., $S_\ell = |B_\ell|^{-1} \int_{B_\ell} S(x)\,dx$. Then, a conservative coarse-grid discretization is obtained by summing a standard single-point upwind discretization for all cells in a block:

$$
\begin{aligned}
(3) \quad S_\ell^{n+1} = S_\ell^n &+ \frac{\Delta t}{\int_{B_\ell} \phi\,dx} \int_{B_\ell} h_S(S^{n+1})\,dx \\
&- \frac{\Delta t}{\int_{B_\ell} \phi\,dx} \Big[f(S_\ell^{n+1}) \sum_{\gamma_{ij} \subset \partial B_\ell} \max(v_{ij}, 0) - \sum_{k \neq \ell} \Big(f(S_k^{n+1}) \sum_{\gamma_{ij} \subset \Gamma_{k\ell}} \min(v_{ij}, 0) \Big) \Big].
\end{aligned}
$$

Notice that if there is bi-directional flow across $\Gamma_{k\ell} = \partial B_k \cap \partial B_\ell$, then the phase-flux across $\Gamma_{k\ell}$ is approximated using both S_ℓ and S_k. Thus, although (3) stems from a single-point upwind scheme on the fine grid, we may obtain a two-sided upwind scheme on the coarse grid. Because the method uses fluxes computed on a finer scale to correctly propagate information over the faces of the coarse grid, it can be viewed as multiscale solver.

The observant reader will notice that the evaluation of the fractional flow function has been moved outside the sum over fine-grid faces in (3). Ideally, the saturation should have been reconstructed in the cells along the block faces $\Gamma_{k\ell}$ to account for subscale variations in the fractional flow. Whereas this is straightforward for an explicit scheme on rectangular blocks, we are not aware of any good method to do so for implicit schemes on arbitrarily shaped blocks. In the following, we will therefore only compute block-averaged saturations.

The multiscale transport solver in (3) has an upscaling counterpart, which only uses one-way fluxes over each coarse interface. These net coarse-scale fluxes are derived by integrating the fine-scale fluxes, giving the following scheme

$$
(4) \quad S_\ell^{n+1} = S_\ell^n + \frac{\Delta t}{\int_{B_\ell} \phi\,dx} \Big[\int_{B_\ell} h_S(S^{n+1})\,dx - \sum_{k \neq \ell} \max\Big(f(S_\ell^{n+1}) \sum_{\gamma_{ij} \subset \Gamma_{k\ell}} v_{ij}, -f(S_k^{n+1}) \sum_{\gamma_{ij} \subset \Gamma_{k\ell}} v_{ij} \Big) \Big].
$$

Like the fine-scale discretization, this is a single-point upwind scheme, but now on the coarse scale. Using net fluxes will simplify the coupling in the resulting nonlinear discrete system. Notice also that the net fluxes used in (4) only coincide with the coarse-grid fluxes computed by the multiscale flow solver when the two solvers are defined over the same coarse grid.

Based on (3) and (4), one can easily develop an adaptive scheme that uses net fluxes across all grid faces where the flux is predominantly unidirectional and fine-scale fluxes across the other faces.

In the next sections, we will study the accuracy (and efficiency) of the two transport solvers described above for various flow-adapted coarse grids. To this end, we will need some more notation. Let S_f and S_c denote the saturation field computed on the fine and coarse grids, respectively, and let w_f and w_c denote the respective water cuts. Moreover, we define $\mathcal{R}$ to be the restriction from the fine to the coarse grid and $\mathcal{P}$ to be the prolongation from the coarse to the fine grid. Finally, we define two different error norms

$$E_s(S_1,S_2) = \frac{1}{T}\int_0^T \frac{\|[S_1(\cdot,t)-S_2(\cdot,t)]\phi(\cdot)\|_1}{\|S_1(\cdot,t)\phi(\cdot)\|_1}\,dt, \qquad E_w(w_1,w_2) = \frac{\|w_1(\cdot)-w_2(\cdot)\|_2}{\|w_2(\cdot)\|_2},$$

where S_1, S_2 and w_1, w_2 denote two different saturation fields and water cuts, respectively. In the following, we will consider three different saturation errors: The projection error $E_s(\mathcal{P}\mathcal{R}S_f, S_f)$ measures the error introduced by representing the saturation on the coarse grid, i.e., the discrepancy between the fine-scale saturation field S_f and the same field restricted to the coarse grid (by averaging all cell values inside each coarse block) and then prolongated back to the fine grid (setting the same average value in all cells inside a coarse block). The other two errors, $E_s(\mathcal{P}S_c, S_f)$ and $E_s(S_c, \mathcal{R}S_f)$, articulate the error introduced by the particular scheme that is used to evolve the coarse-scale saturations, measured as the discrepancy between S_f and S_c on the fine grid and on the coarse grid, respectively.

3. Flow-Based Nonuniform Coarsening

We will use the nonuniform coarsening method by Aarnes et al. [2] as our starting point for developing efficient transport solvers on flow-adapted coarse grids. This method partitions the fine grid into coarse blocks according to flow magnitude by separating regions of high and low flow. The algorithm follows four steps:

(1) *Compute an initial partitioning.* To this end, we use the logarithm of the flow as our indicator function, $I(c_i) \propto \log|\vec{v}(c_i)|$, which is segmented into ten uniform bins: $c_i \subset \tilde{B}_\ell$ if $I(c_i) \in [I_\ell, I_{\ell+1})$. Each bin $\tilde{B}_\ell$ may consist of a multiply connected set of cells and must be postprocessed and split into singly-connected blocks.
(2) *Merge small blocks.* If a block B' has too small volume, $|B'| < \frac{N_L}{n}|\Omega|$, it is merged with the neighbouring block B that has the closest I-value defined as $I(B)|B| = \int_B I(c)\,dx$.
(3) *Refine blocks with too much flow.* If $\int_B I(c)\,dx > \frac{N_U}{n}\int_\Omega I(c)\,dx$, then
   (a) Pick an arbitrary cell c_0 belonging to ∂B.
   (b) Find the cell $c_1 \subset B$ that is furthest away from c_0, using, e.g., the distance between the cell centroids as a metric, and define $B' = c_1$.
   (c) Progressively enlarge B' by adding cells surrounding B'; that is, add $c \subset \mathcal{N}(B')$ if $c \not\subset B'$, as long as $\int_{B'} I(c)\,dx \le \frac{N_U}{n}\int_\Omega I(c)\,dx$.
   (d) Define $B = B \setminus B'$ and continue to refine B if the upper bounds are still violated.
(4) *Repeat Step 2.*

The four steps are illustrated in Figure 2. The nonuniform coarsening method may be viewed as a heuristic algorithm that seeks to: (i) minimize the heterogeneity variation inside each grid block, using $I(c)$ as a measure of heterogeneity, and (ii) equalize the total flow through each block, given a lower constraint on block volumes. The latter condition is inspired by Lagrangian grids formulated using streamline coordinates; as discussed in [2]. Steps 2 and 3 make up the first loop in an iterative method that may or may not converge to a unique and optimal grid. In practise, a single (or one and a half) iteration is sufficient to generate a good grid.

The shape of the blocks is determined by the neighbourhood definition $\mathcal{N}(c)$, so that the algorithm creates coarse grids with block boundaries that are aligned with distinct features in

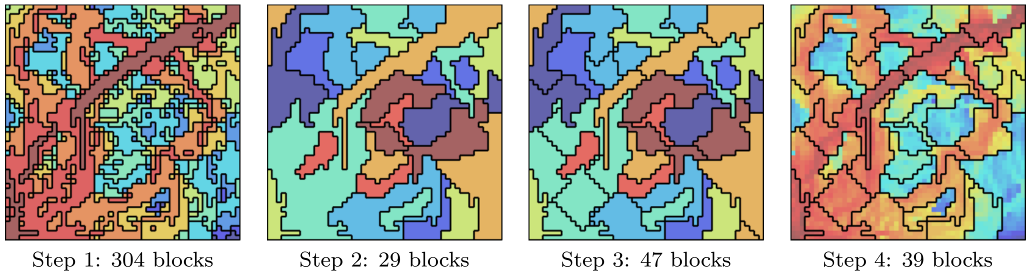
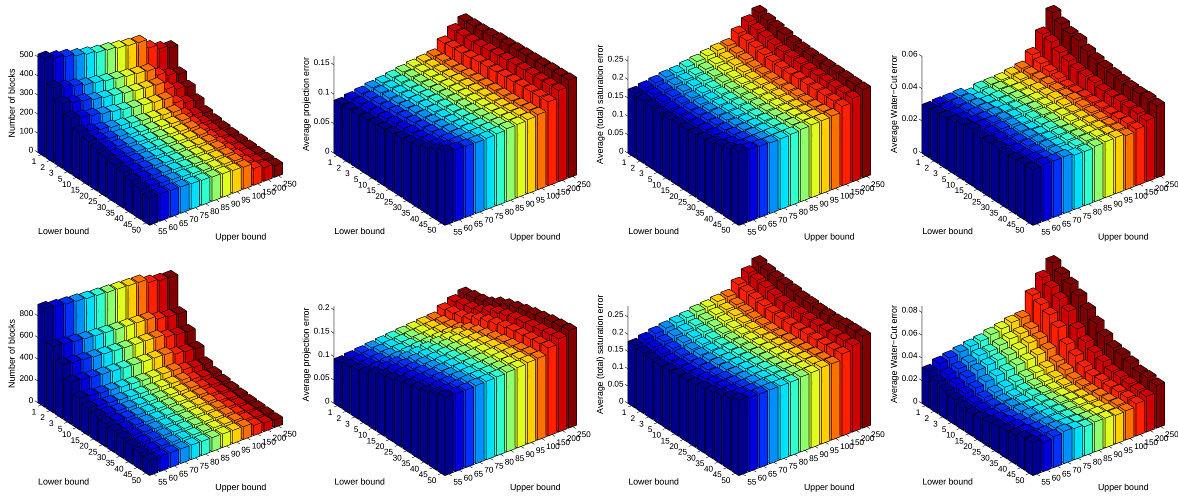


FIGURE 2. Illustration of Steps 1 to 4 for a quarter five-spot simulation of a 50×50 excerpt from Layer 68 from Model 2 of the SPE10 benchmark [5]. The coarsening parameters are $N_L = 20$ and $N_U = 100$. In the rightmost plot, the colours represent logarithm of the underlying fine-scale velocity field.

the permeability field as reflected in the flow indicator $I(c)$. In the original algorithm, $\mathcal{N}(c)$ is defined as the face neighbours: that is, cells c and $\tilde{c}$ are neighbours if they share a common face. For a Cartesian grid, this corresponds to the usual 5-point neighbour relation, and gives rise to the characteristic diamond-shaped cells seen in the two rightmost plots of Figure 2. To get blocks with a more regular shape, we will in the following use a 9-point neighbour relation (for logically Cartesian grids). We refer the reader to [11] for a more thorough discussion of neighbour relations in coarsening algorithms.

The algorithm presented above is fully automated in the sense that the degree of coarsening is determined by two user-supplied parameters: N_L gives a lower bound on the volume of blocks and prevents the algorithm from generating too small blocks, and N_U gives an upper bound on the total amount of flow through each grid block and prevents the algorithm from generating too large blocks. The parameters N_L and N_U act as soft lower and upper constraints on the number of cells in the coarse blocks. To see this, consider a grid in which all cells have the same volume (Ω/n) and the same indicator value I_c. Then, if n_B denotes the number of cells in block B, the lower bound simplifies to $n_B > N_L$ whereas the upper bound simplifies to $n_B \leq N_U$. In the general case, N_L and N_U can only be used as indicative values and should be chosen somewhat apart to give the algorithm necessary flexibility. For practical applications, it is important that the method is robust with respect to the input parameters. In our experience, the number of grid block and the quality of the grid are not very sensitive to different choices of N_L and N_U, as illustrated in the example below.

Example 1 (Layers of SPE10)**.** *We consider a quarter five-spot well pattern on a regular Cartesian grid with petrophysical data sampled from individual layers of the SPE10 benchmark model [5], which contains* $220 \times 60 \times 85$ *cells and consists of two formations: Layers 1–35 represent a shallow-marine Tarbert formation whereas Layers 36–85 represent a fluvial Upper Ness formation. Figure 3 shows the number of blocks, projection error, total saturation error, and water-cut error for all layers in the Tarbert formation and all layers in the Upper Ness formation (Layers 74–76 were excluded because of long simulation times caused by small porosity values). For both formations, the number of coarse blocks is mainly determined by* N_L*, in particular for the fluvial layers. For the smooth Tarbert permeability, the projection error is dominated by the size of the (largest) coarse blocks: the error increases with increasing values of* N_U *but is almost constant with respect to* N_L*. Interestingly, the lowest saturation errors are observed for* intermediate *values of* N_L *and small values of* N_U*. Moreover, for large values of* N_U*, the water-cut errors decrease with increasing values of* N_L*. Conversely,*



number of blocks | projection error $E_s(\mathcal{PR}S_f, S_f)$ | total saturation error $E_s(\mathcal{P}S_c, S_f)$ | water-cut error, $E_w(w_c, w_f)$

FIGURE 3. The nonuniform gridding method applied with varying parameters N_L and N_U on layers of the SPE10 model. In the upper row, the histogram values are averaged over the Tarbert formation and in the lower row over Layers 36–73 and 77–85 in the Upper Ness formation.

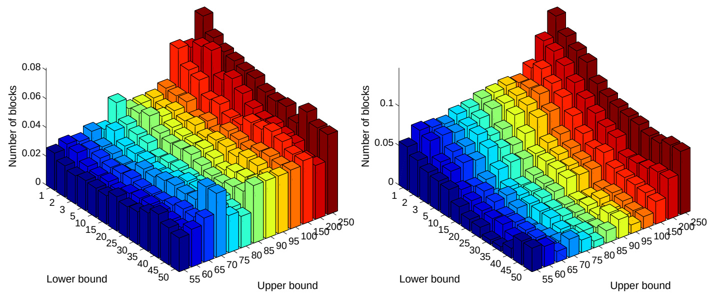


FIGURE 4. Water-cut errors for the nonuniform coarsening method applied with varying parameters to Layer 28 (left) and Layer 48 (right) of SPE10.

for fixed N_L, the water-cut errors increase with N_U, because larger upper bounds allow larger blocks, which in turn cause smearing of the saturation fronts and typically lead to earlier water breakthrough. For smooth heterogeneities, one should therefore try to minimize the variation in block sizes.

For the channelized permeability, the saturations will follow narrow flow channels and the projection error is dominated by how well the grid resolves these channels; N_L is the decisive parameter to this end. This is confirmed by the histogram in which the largest increase in projection error is observed in the direction of increasing N_L values. The water-cut error, on the other hand, decreases with increasing N_L values. A possible explanation is that the smallest blocks occur in and near the high-permeable channels and as the sizes of these blocks increase, the blocks adapt to the high-flow paths and thereby improve the grids' ability to correctly evolve saturations.

Finally, we notice the existence of relatively flat regions in the middle of the parameter domain, where reasonable accuracy is achieved in a robust manner. This is not an averaging effect—similar regions are observed when considering a single layer. However, for a single layer, small changes in parameters can sometimes lead to disproportionate changes in accuracy as shown in Figure 4 for the water-cut errors in Layers 28 and 48.

For highly heterogeneous reservoirs, and in particular for strongly channelized reservoirs as seen above, the nonuniform coarsening method produces grids that capture the dominating flow patterns very accurately, even for high upscaling factors. This was demonstrated in [2], both for structured and unstructured grids, and particularly for the fluvial part of the widely used SPE10 model [5], for which flow-based coarsening significantly reduces saturation errors and errors in water cuts compared with a regular Cartesian coarsening. In [2], the flow indicator $I(v)$ was computed from a single-phase flow driven by a particular well configuration and set of flow rates. The authors presented a few numerical experiments in which grids created from one well configuration were used to simulate the flow resulting from other and different well patterns to show that the method can be robust with respect to varying flow patterns as long as the flow field is dominated by the underlying heterogeneity.

Generally, our experience is that the nonuniform coarsening algorithm is not as robust, accurate, and efficient as it was somewhat optimistically reported in [2]. On the upper layers of the SPE10 model (the somewhat smoother Tarbert formation), the flow-based grids do not give better accuracy than a straightforward Cartesian coarsening. In fact, we have run numerous experiments with varying coarsening factors for the SPE10 and other models that all indicate that Cartesian coarsening predominantly gives slightly better accuracy in saturation fields and water cuts than flow-based grids for cases with small or moderate heterogeneity. Likewise, one can construct cases in which (more) significant changes in well configuration and flow rates take away the advantages of flow adaption. Finally, flow-based grids typically have more irregular blocks with more neighbours and coarse-block interfaces and this tends to increase the coupling in the discrete nonlinear system. The number of couplings in the nonlinear system will typically affect how costly it is to solve, and hence it is desirable to increase the regularity of the blocks if this does not significantly affect the ability to resolve flow patterns; we will come back to this discussion in Section 5.

Altogether, our experiments suggest that the original algorithm, as presented above, has a tendency of exaggerating the effect of the underlying velocity pattern and thus creating grids that are more irregular and have larger differences in block sizes than what is needed. In the next section, we therefore discuss how to reduce the degree of flow adaption and demonstrate how flow adaption can be combined with *a priori* partitions to increase the gridding flexibility and better balance the influence of the underlying heterogeneity on the gridding process.

4. Imposing *A Priori* Partitions

In many cases, a standard grid may work quite well and the user may only want a limited degree of flow adaption to create a simpler and less regular grid. In particular, the user may possess expert knowledge of what has the most influence on accuracy and may want, e.g., to impose *a priori* information on the (local) shape of the coarse blocks. Likewise, there can be other geological features that the user may want to utilize to create grids that better adapt to the underlying geology or serve other purposes such as reducing the need for multiphase upscaling. In this section, we will therefore demonstrate how to use such *a priori* information to create grids that are more feasible and/or give improved accuracy.

First of all, we propose an additional step in the original algorithm that consists of intersecting the initial flow-based colouring of cells in the first step with an *a priori* partitioning. This intersection will then be the basis for the rest of the steps, which remain unchanged from the original algorithm. We point out that this additional step is applicable to *any* grid as long as the user is able to specify an *a priori* partition vector p_a. Second, to reduce the influence of the underlying heterogeneity on the coarsening, we look at the initial colouring of cells. Reducing the number of bins means that we increase the size of the blocks resulting from the flow-based colouring, and hence to a large extent preserve the *a priori* partition. If the sizes of the *a priori* blocks are within the bounds specified by the N_L and N_U parameters, a large number of these blocks will be left intact by Steps 2–4.

To illustrate the effect of the extra step and the adaptive number of initial bins, we will consider an example in which we seek to impose a regular Cartesian partitioning on the flow-based gridding process.

Example 2 (Layer 1 of SPE10)**.** *The leftmost plot in Figure 5 shows the intersection of a regular* 6×22 *Cartesian coarsening with the initial flow-based colouring. Altogether, this generates a finer partitioning as the starting point of the merging and refinement steps. The added interfaces are straight lines that will typically be preserved in low-flow regions, as seen by comparing Grid 1 with the grid generated by the original algorithm. In the original algorithm and for Grid 1, we have used 10 bins in the initial colouring. This amounts to approximately one bin per order of magnitude in the underlying velocity field for the fluvial parts of the SPE10 model. For the layers in the Tarbert formation, as considered here, the logarithmic span in the velocities is significantly smaller. For Grid 2, we therefore have chosen four, rather than ten, initial bins because the logarithmic velocity span equals 4.4 for Layer 1. As a result, more blocks from the* a priori *partitioning remain unchanged throughout Steps 2–4, and the resulting grid has a much more regular structure than Grid 1. On the other hand, for both Grid 1 and Grid 2, we observe that the high-flow channels are distinctly outlined and correspond to the ones detected by the original algorithm. In regions of low flow, we also observe that the original grid has a much more complex grid structure, with a larger number of neighbouring connections.*

We have run a large number of different studies using the original and our improved algorithm (a few quantitative results will be reported in the next section). Choosing the number of bins according to the logarithm span in velocity (or possibly permeability) seems to be a good choice. The other parameters, however, must be chosen with some care and possibly be fine-tuned to give optimal results. For small and moderate heterogeneities, it seems particularly important to balance the choice of the two partitioning mechanisms. Choosing a coarse *a priori* partitioning and small partitioning parameters (N_L, N_U), means that the flow-based partitioning will dominate and any advantages from the *a priori* partitioning disappears. Likewise, choosing a fine *a priori* partitioning and large values for N_L and N_U, implies that most cells are merged into large blocks in Step 2 and then refined in Step 3. The resulting grid will hence have the characteristics of the original algorithm, and the effects of the *a priori* partitioning disappears.

A priori information can also be used to distinguish different geological features that need to be taken into account and/or preserved during the coarsening. In the next example, we demonstrate how one can use facies numbers as the initial partitioning to ensure that the coarse blocks do not cross facies boundaries. This can be very useful if the facies have different relative permeabilities or capillary pressure curves. By making sure that each coarse block

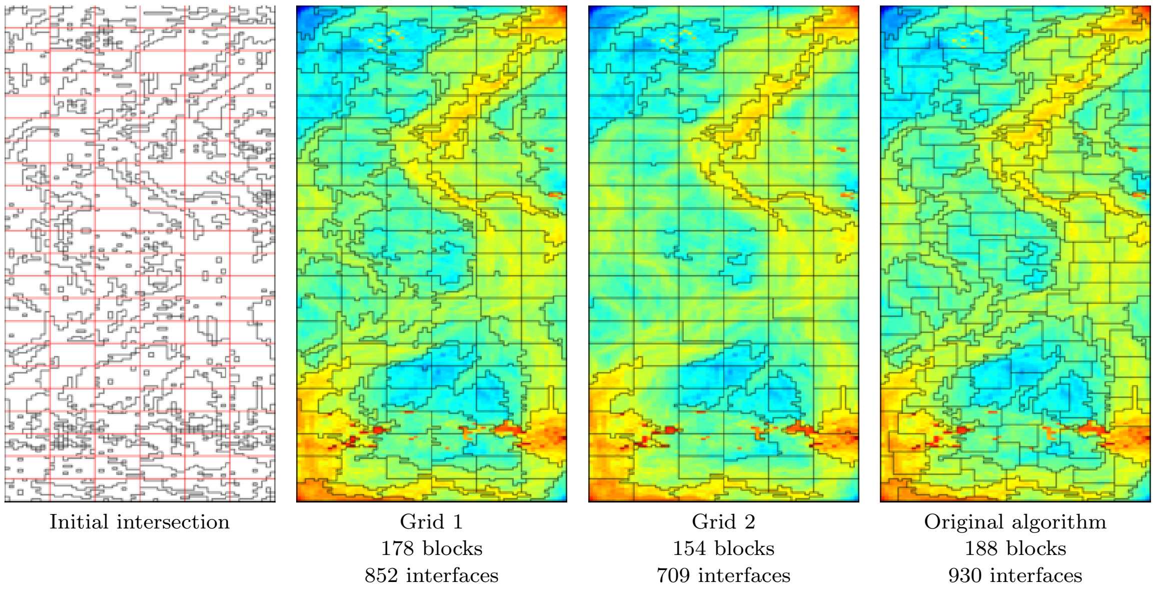
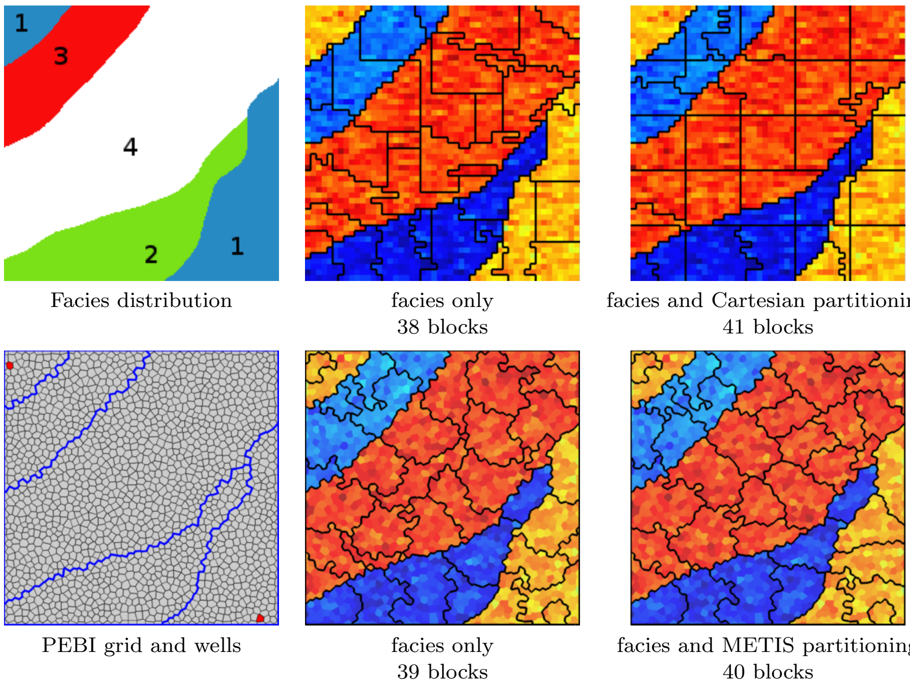


FIGURE 5. Nonuniformly coarsened grids for Layer 1 of Model 2 from SPE10. The leftmost plot outlines the intersection of the initial partitioning, where red lines represent the *a priori* partitioning and black lines the partitioning into bins. Grid 1 is generated with 10 initial bins and Grid 2 with the number of bins adapted to the flow, and for both we used coarsening parameters N_L=25, N_U=125 and a uniform 6×22 *a priori* partitioning. The rightmost plot shows the grid generated by the original algorithm with parameters $N_L = 15$, $N_U = 75$ and a 9-point neighbourhood relation. All three grids are outlined on top of the logarithm of the fine-scale velocity field, computed by solving a quarter five-spot problem.

consists of only one facies, one can avoid cumbersome upscaling of relative permeabilities and capillary pressure functions to the coarse grid.

Example 3 (Facies model)**.** *We consider a rectangular domain with a facies distribution as shown in the upper-left plot of Figure 6. The permeability distribution follows a lognormal distribution inside each facies with mean values of* 400*,* 20*,* 35*, and* 800 *mD, respectively. The permeability distribution is sampled on a regular* 50×50 *Cartesian grid. The reservoir is produced by an injector-producer pair located near the upper-left and lower-right corners of the domain, respectively; the wells are shown as red dots in the lower-left plot in Figure 6. For illustration purposes, we generate two coarse grids: for the first one, we only use the facies distribution as our* a priori *partition vector, and for the second one, we impose a regular Cartesian partitioning, in addition, as discussed for Grid 2 in Example 2. The two grids are shown in the middle and right plot in the upper row of Figure 6. From the plots, we clearly see that the coarse blocks are confined inside a single facies. By additionally imposing an* a priori *Cartesian partitioning, we increase the number of blocks, but also get more regularity in our coarsening.*

To demonstrate that our method is not restricted to Cartesian grids, we consider the same facies and permeability distribution (re)sampled on the PEBI grid (unstructured Voronoi grid) with 1697 cells shown in the lower-left plot of Figure 6. The middle and right plots in the lower row of Figure 6 show two coarse grids by our algorithm. The difference in the two



Facies distribution — facies only, 38 blocks — facies and Cartesian partitioning, 41 blocks

PEBI grid and wells — facies only, 39 blocks — facies and METIS partitioning, 40 blocks

Figure 6. Coarsening using *a priori* information of facies distribution. In the upper row, the underling fine grid is a regular 50×50 Cartesian grid. In the lower row, the fine grid is a fully unstructured PEBI grid with 1697 cells.

grids lie in how we perform the refinement in Step 3 of the algorithm. In the middle plot, we have used the refinement method outlined above with the unstructured equivalent of the 9-point neighbour definition. Also in this case, the blocks are clearly confined to a single facies, as desired. However, because of the unstructured connections in the fine-scale grid, the coarse blocks are significantly more irregular than in the Cartesian case. In the right plot, we have instead used the software package METIS [14] to perform the refinement step, this in an attempt to improve the regularity of the blocks.

Although the results presented in the previous example are encouraging for PEBI grids, we believe that more research is needed to come up with a better algorithm for refining large blocks in Step 3, particularly in 3D, in a way that imposes more regularity on the resulting blocks. The same problem arises when generating coarse grids to be used in multiscale flow solvers. Because of the complex connection pattern of the underlying unstructured grid, it is generally difficult to come up with coarse blocks that do not have irregular boundaries. One possibility would be to include a smoothing step, but this would need to be carefully designed to distinguish between irregular faces caused by flow adaption and irregular faces induced by the unstructured connections.

In the next example, we will study a realistic 3D model represented in the industry-standard corner-point format, i.e., as a grid that consists of a set of hexahedral cells that are topologically aligned in a Cartesian fashion so that the cells can be numbered using a logical ijk index. From a coarsening perspective, the underlying ijk index is very useful and can e.g., be utilized to impose a regular *a priori* partitioning as in Example 2. Here, however, we will use saturation regions as our *a priori* partitioning.

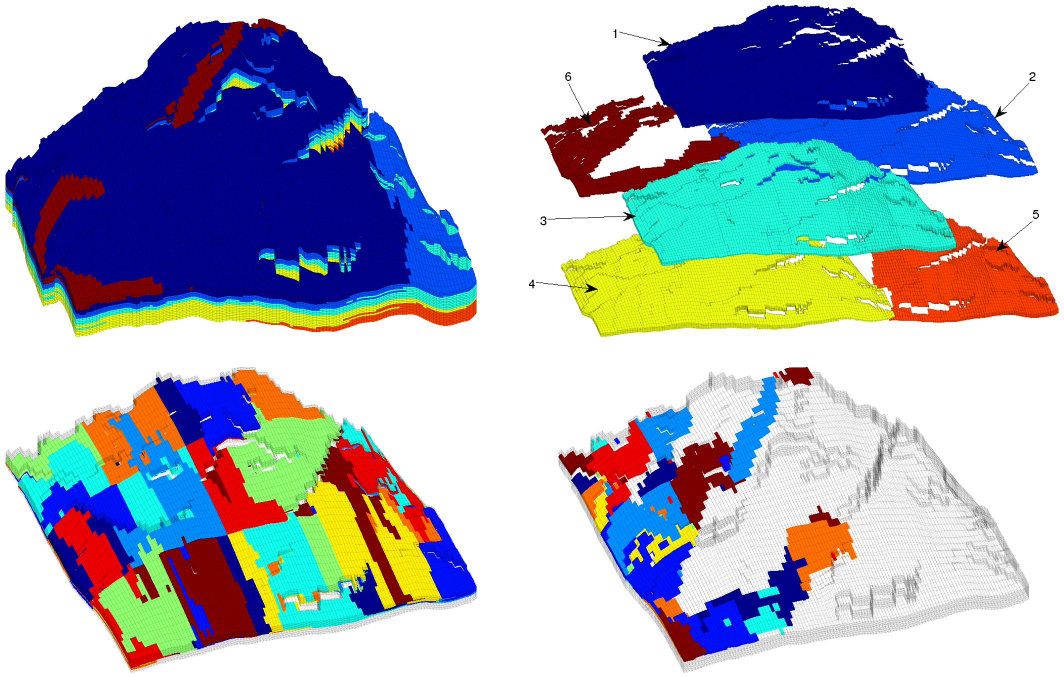


Figure 7. Flow-based coarsening of a faulted SAIGUP model. The upper-left plot shows the structural model. The upper-right plot shows the six saturation regions. The lower plots show the coarse-grid blocks in saturation regions three and six, respectively.

Example 4 (SAIGUP)**.** *We consider one of the faulted models from the SAIGUP study [18], which we have already used in Figure 1 to illustrate the key steps of a multiscale flow solver. The petrophysical parameters for the model were generated on a regular* $40 \times 120 \times 20$ *Cartesian grid and then mapped onto a structural model described using the corner-point grid format. The left plot in Figure 7 shows the structural model. The colours represent the six different saturation regions (Eclipse keyword SATNUM), which may or may not correspond to different facies or rock types. Because the main purpose of the example is to illustrate the gridding capabilities on a model with realistic geometry and petrophysical properties, we use a simple injector–producer pair (see Figure 1) and create a relatively coarse flow-based grid.*

The coarse grid was created by imposing the six saturation regions as an a priori *partitioning. Moreover, in Steps 2 and 4, we restricted the neighbourhood definition to only include cells that were part of the same saturation region. As we see from Figure 7, the coarse blocks have complicated shapes but seem to follow the saturation regions; this is particularly evident in region six. The plot may be slightly deceiving with respect to the connection between blocks: blocks that appear to be multiply connected are, in fact, singly connected through cells in deeper layers that are not visible in the plot.*

Another example of flow-based gridding on corner-point grids was presented by Krogstad et al. [16], who used such grids to accelerate forward simulations in a production optimization workflow. In the next section, we will give a more quantitative study of the gridding methods introduced in the previous section when applied together with the multiscale transport solver (3) and the coarse-scale solver (4).

5. Multiscale versus Coarse-Scale Solver

First, we start by discussing the computational efficiency. Both discretizations, (3) and (4), lead to a system of nonlinear equations that will typically be solved by a Newton–Raphson type method. The computational efficiency of the nonlinear solver will to a large extent depend on the structure and the condition number of the system. The best we can hope for, is that the discrete system has an upper (or lower) triangular form, because then we can use a nonlinear Gauss–Seidel solution procedure and compute the unknown block saturations by backward (or forward) back-substitution. This is clearly possible for a 1D problem. Likewise, in the absence of gravity and capillary forces, (2) has an inherent causality principle that is utilized in streamline methods to transform the multi-dimensional transport equation into a family of 1D problems along streamlines.

Natvig and Lie [19] recently demonstrated how this causality principle can be used to compute an optimal flow-based ordering that renders the system in a block-triangular form. This is achieved by topological sorting of the directed graph formed by setting the grid blocks as vertices and the discrete fluxes as edges. If the flow solver is monotone, the directed graph is acyclic and the reordered system will have a completely triangular form, so that the saturation in each grid block only depends on the saturation values in the block's upstream neighbours. Hence, the solution can be computed block-by-block, moving gradually downstream from wells or other fluid sources, solving a single *scalar* equation in each block (for two-phase flow). For non-monotone (and multiscale) flow solvers, there will be some circulation in the discrete fluxes, which will lead to larger matrix blocks that contain grid blocks that are circularly dependent. Still, the system can be solved by an efficient block-wise back-substitution procedure, in which the circularly dependent grid blocks are solved for simultaneously. Furthermore, if the linearization is performed *locally* on each matrix block, we gain local control over the nonlinear (Newton–Raphson) iterations and thereby obtain highly efficient and (near) optimal nonlinear solvers [19, 20]. Henceforth, we will refer to grid blocks that only depend on their upstream neighbours as *scalar components*, whereas circularly dependent grid blocks will be called *connected components*. Finding the connected components and the optimal ordering are standard and efficient $\mathcal{O}(N)$ operations from graph theory that are easy to implement.

Rather than considering a specific nonlinear solver, we will in the following use the degree to which the discrete system can be reordered into a triangular form as a measure of the efficiency of the grid. Our idea is that if such a structure exists, any efficient nonlinear solver should ideally be able to exploit it. The efficiency of the nonlinear solver will therefore depend on the size of the largest block system, because solving an $n \times n$ nonlinear system using the Newton–Raphson method will in most cases be significantly more expensive than solving n scalar nonlinear problems, which can be solved very efficiently using e.g., Ridder's method as discussed in [19].

Example 5 (Layers 1 and 37 of SPE10)**.** *Continuing from Example 2, we consider a quarter five-spot problem on two layers from the SPE10 model: Layer 1 from the Tarbert formation and Layer 37 from the fluvial Upper Ness formation. For each layer, we will use three different grids: a* 10×22 *Cartesian grid, a grid generated by the original algorithm from [2], and Grid 2 from Example 2 and its equivalent on Layer 37. Table 1 reports the corresponding number of scalar components, number of connected components, number of blocks in the largest connected component, and number of off-diagonal entries.*

TABLE 1. Comparison of the matrix structure on two layers of SPE10. For each column, the first number refers to the multiscale solver (3) and the second number refers to the coarse-scale solver (4).

	Layer 1						Layer 37					
	Grid 2		Original		Cartesian		Grid 2		Original		Cartesian	
# grid blocks	154	154	188	188	220	220	169	169	205	205	220	220
# scalar components	1	127	0	138	75	220	1	65	0	100	16	208
# connected components	4	7	1	7	50	0	1	13	2	9	24	3
largest component	89	6	188	26	10	–	168	33	203	26	48	4
# off-diagonal elements	507	342	798	495	503	408	756	439	911	557	574	408

Let us first look in detail at a few of the grids. We start with the Cartesian grid for Layer 1. Using bi-directional fluxes, there are 50 connected components that contain at most ten grid blocks and 75 scalar components. If we instead use net fluxes, there are only scalar components in the system, which can therefore be solved one grid block at the time. Next we consider Grid 2 on Layer 37. Figure 8 shows the concept of the matrix reordering for the case with net fluxes: Out of the 169 blocks in the grid, 104 blocks have some circular dependence and are part of thirteen connected components: two large, one intermediate, and ten small. The largest component contains 33 blocks, in which the saturations must be computed simultaneously by solving a 33×33 *nonlinear system. Similar block systems must be solved for the other twelve connected components. The remaining 65 scalar components are only connected to their upwind neighbours and here the saturation can be computed by solving a scalar nonlinear problem once the upwind neighbours have been computed.*

Overall, we see that the use of net fluxes, as in (4)*, rather than bi-directional fluxes across the block interfaces, as in* (3)*, reduces the number of off-diagonal elements, the number of connected components, and the size of these components on all six grids and hence leads to a nonlinear system that generally will be less expensive to solve. This conclusion should also be true in the general case: replacing bi-directional fluxes with net fluxes will decrease the number of couplings in the nonlinear discrete system and hence decrease the computational cost.*

Let us now briefly look at the accuracy of the two schemes, (3) *and* (4)*. In Figure 9, we compare the corresponding coarse-scale saturations with the fine-scale reference saturation for the original flow-based grid on Layer 37. In the visual norm, the coarse-grid solver appears to resolve the saturation field more accurately than the multiscale solver: the solver exhibits less numerical diffusion in low-flow regions and has been able to capture pockets of bypassed oil. We believe that the difference in the two solvers can be explained as follows: the multiscale solver effectively uses central differences and hence spreads small saturations induced by numerical diffusion into large(er) areas. The coarse-scale solver is effectively an upwind solver and hence has less coarse-scale numerical diffusion.*

In a multiscale setting, the results above are interesting for the following reason. Krogstad et al. [16] have previously demonstrated that the combination of a multiscale flow solver and a flow-adapted grid can be very efficient if one can avoid communication through the underlying fine grid and only store fine-scale fluxes at the coarse interfaces and use precomputed mappings to move saturations from the transport grid to the multiscale flow solver. Replacing fine-scale fluxes on each coarse-grid interface by net fluxes will decrease the communication need even further and hence increase the efficiency of the overall solver.

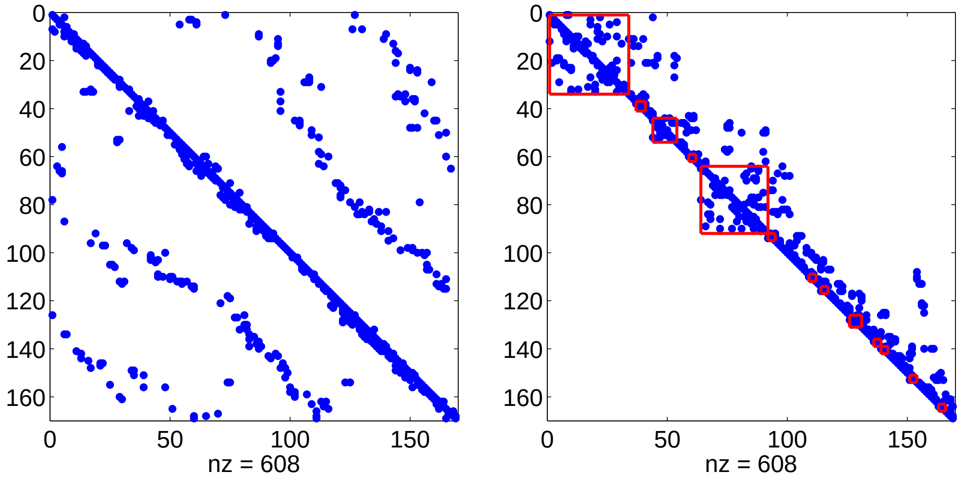


FIGURE 8. Matrix structure of the flux vector for Grid 2 on Layer 37 with the coarse-grid solver (4). The left plot shows the matrix with the ordering coming from the coarsening algorithm, whereas the right plot shows the matrix structure after we have performed a flow-based reordering. Altogether, 65 of the saturation values depend only on their upwind neighbours, whereas the remaining 104 have some circular dependence and are part of one of the thirteen connected components. The corresponding matrix blocks are marked in red: two large, one intermediate, and ten small.

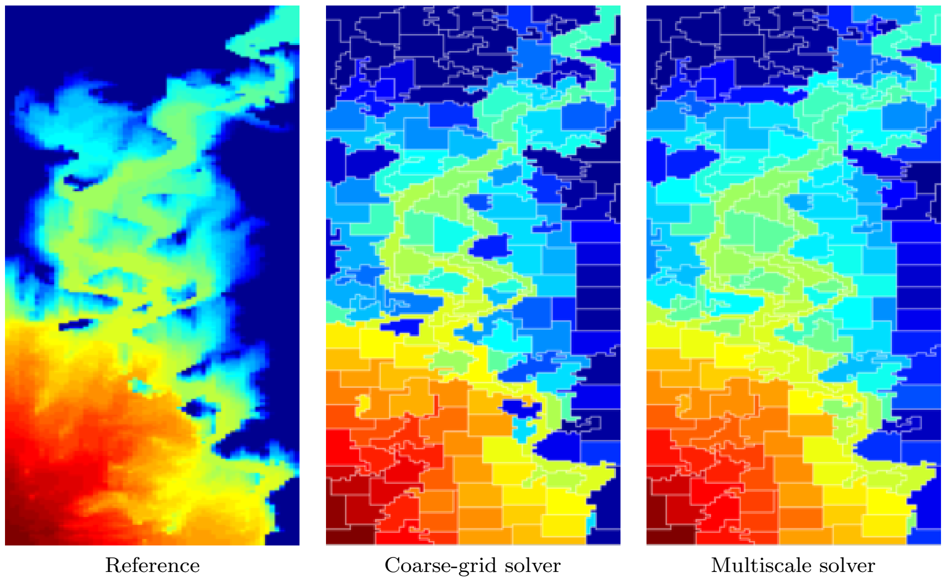


FIGURE 9. Comparison of saturations at dimensionless time 0.8 PVI for the coarse grid generated by the original algorithm on Layer 37.

Table 2. Comparison of saturation and water-cut errors up to dimensionless time $T = 1.0$ PVI for quarter five-spot simulations on the individual layers of the SPE10 model. The upper part of the table shows errors for the multiscale solver (3), the middle part shows errors for the coarse-scale solver (4), and the lower part gives statistics for the grids.

	Tarbert formation				Upper Ness formation			
	Grid 1	Grid 2	Original	Cartesian	Grid 1	Grid 2	Original	Cartesian
$E_s(\mathcal{PR}S_f, S_f)$	0.0920	0.0941	0.1042	0.0911	0.1394	0.1371	0.1355	0.1772
$E_s(\mathcal{P}S_c, S_f)$	0.2071	0.1910	0.2426	0.1687	0.2180	0.2124	0.2243	0.2305
$E_s(S_c, \mathcal{R}S_f)$	0.1784	0.1599	0.2100	0.1381	0.1572	0.1522	0.1683	0.1604
$E_w(w_c, w_f)$	0.0649	0.0695	0.0773	0.0701	0.0613	0.0609	0.0668	0.0982
$E_s(\mathcal{P}S_c, S_f)$	0.1591	0.1607	0.1875	0.1619	0.1827	0.1795	0.1862	0.2191
$E_s(S_c, \mathcal{R}S_f)$	0.1220	0.1237	0.1459	0.1302	0.1155	0.1135	0.1225	0.1486
$E_w(w_c, w_f)$	0.0349	0.0473	0.0444	0.0647	0.0232	0.0237	0.0325	0.0844
# blocks: span	232–268	217–261	233–312	264	202–234	205–241	220–303	264
# blocks: mean	249	236	275	264	216	222	264	264
# faces: mean	1175	1069	1363	1090	1049	1070	1309	1090

In the next example, we will study the accuracy of the multiscale and the coarse-scale transport solvers on two-dimensional horizontal cross-sections of the SPE10 model [5].

Example 6 (All layers of SPE10). *For each layer of the SPE10 model, we conduct a quarter five-spot simulation using a uniform* 12×22 *Cartesian coarsening as well as the three coarsening choices discussed in Example 2. The Cartesian grid corresponds to an upscaling factor of 50 and the parameters in the flow-based algorithm were hence chosen to produce a similar (or slightly larger) upscaling factor. Table 2 shows a comparison of water-cut and saturation errors, where the saturation errors have been split in three parts: error measured on the fine grid, error measured on the coarse grid, and projection error. All simulations used 20 equally-spaced pressure steps with 15 equally-spaced substeps in the transport solvers. To make a consistent comparison of the fine-scale and coarse-scale transport solvers, the pressure updates were performed on the underlying fine grid also for the coarse-grid transport solver.*

We start by considering the original multiscale transport solver (3) *from [2]. Here, the results in the upper part of the table clearly show that the water-cut and projection errors are significantly* reduced *on the fluvial layers by using the flow-based coarsening methods, whereas there is no significant change in the water-cut error and slightly larger projection errors on the smooth Tarbert layers. On the other hand, comparing with the Cartesian grid, we see that the coarse-scale evolution error is significantly* increased *on the Tarbert layers because of increased numerical dissipation near the largest blocks, and slightly reduced on the fluvial layers because high-flow channels are more accurately represented. Finally, we notice that both Grid 1 and Grid 2 consistently produce lower errors and fewer blocks and faces than the original coarsening algorithm.*

Looking at the coarse-scale scheme (4)*, we see that this solver consistently gives lower errors than the multiscale solver for all four grids, in particular for the water-cut error. As discussed in Example 5, we believe this can be attributed to a significantly lower coarse-scale diffusion. Moreover, on the Tarbert layers, the errors for Grid 1 and Grid 2 are lower than for the regular Cartesian grid.*

Having presented the results in the previous example, we must concede that the results are slightly volatile. Based on a large number of experiments using the multiscale transport solver, we see that using a different upscaling factor can in many cases produce more favourable results for the flow-based coarsening methods (on the Tarbert layers), but may also in certain cases produce slightly worse results. The coarse-scale solver is more recent, and we have not yet conducted an equally extensive study. Still, we believe that this solver will prove to be more accurate because the single-point flux approximation generally has less coarse-scale diffusion than the bi-directional flux.

6. Dynamically Adaptive Grid

For displacements with strong displacement fronts, the majority of the projection error, which was briefly discussed in Example 6, is associated with inaccurate representation of the fluid front. Think of a typical Buckley–Leverett profile: in the unswept area ahead of the displacement front, the solution is constant and can be accurately represented on a relatively coarse grid. Likewise, behind the displacement front, the solution is smooth and slowly varying and can hence be evolved on a coarse grid. In the absence of capillary forces, or other second-order terms in the transport equation (2), the displacement front is a discontinuity that needs high grid resolution to be accurately approximated. Motivated by these observations, we will in this section demonstrate how the simulation accuracy can be significantly improved by dynamically adding local resolution near strong saturation fronts. Somewhat similar ideas were used by Lee et al. [17] and Zhou et al. [22] in their adaptive multiscale finite-volume method.

Because all grids considered herein are obtained by coarsening an underlying fine grid, it is relatively straightforward to add local refinement by manipulating the partition vector p, giving a local resolution that may be less or equal that of the fine grid. Moreover, this refinement can be added or removed dynamically provided we have good indicators of when to do so. Herein, we rely on the simplest approach possible, namely to compute each saturation step twice: once with a coarse resolution to estimate the movement of the front and once (locally) with higher resolution to resolve the movement more accurately. After the first step, we mark all blocks in which the saturation change from the previous time step exceeds a prescribe tolerance. These blocks are then refined. Likewise, after the second saturation step, we go through all refined blocks and mark those where the total fine-scale saturation changes are below another prescribed tolerance. In each marked block, the saturations are averaged back onto the original coarse block, and the refinement is removed by manipulating the partition vector.

Example 7 (Layers 1 and 37 from SPE10)**.** *We revisit the two models studied in Example 5. In the fluid model we use quadratic relative permeabilities with a viscosity of 1 cP for the displacing fluid (water) and a viscosity of 0.2 cP for the displaced fluid (oil). This favourable displacement ratio will lead to a sharp front, for which local refinement of the grid is needed to avoid excessive numerical smearing when using a relatively coarse grid.*

Figure 10 shows the solution at dimensionless time 0.5 PVI computed on a coarse grid (Grid 2 from Figure 5), a coarse grid with local adaptive refinement, and on the original 60×220 *Cartesian grid using one pressure solve and sixty-five equally-spaced transport steps. For Layer 1, we notice the excessive smearing at the fluid front, midway through the reservoir, but also that the coarse grid completely fails to capture the pocket of bypassed oil. The adaptive grid, on the other hand, is in good correspondence with the fine-scale reference solution. For*

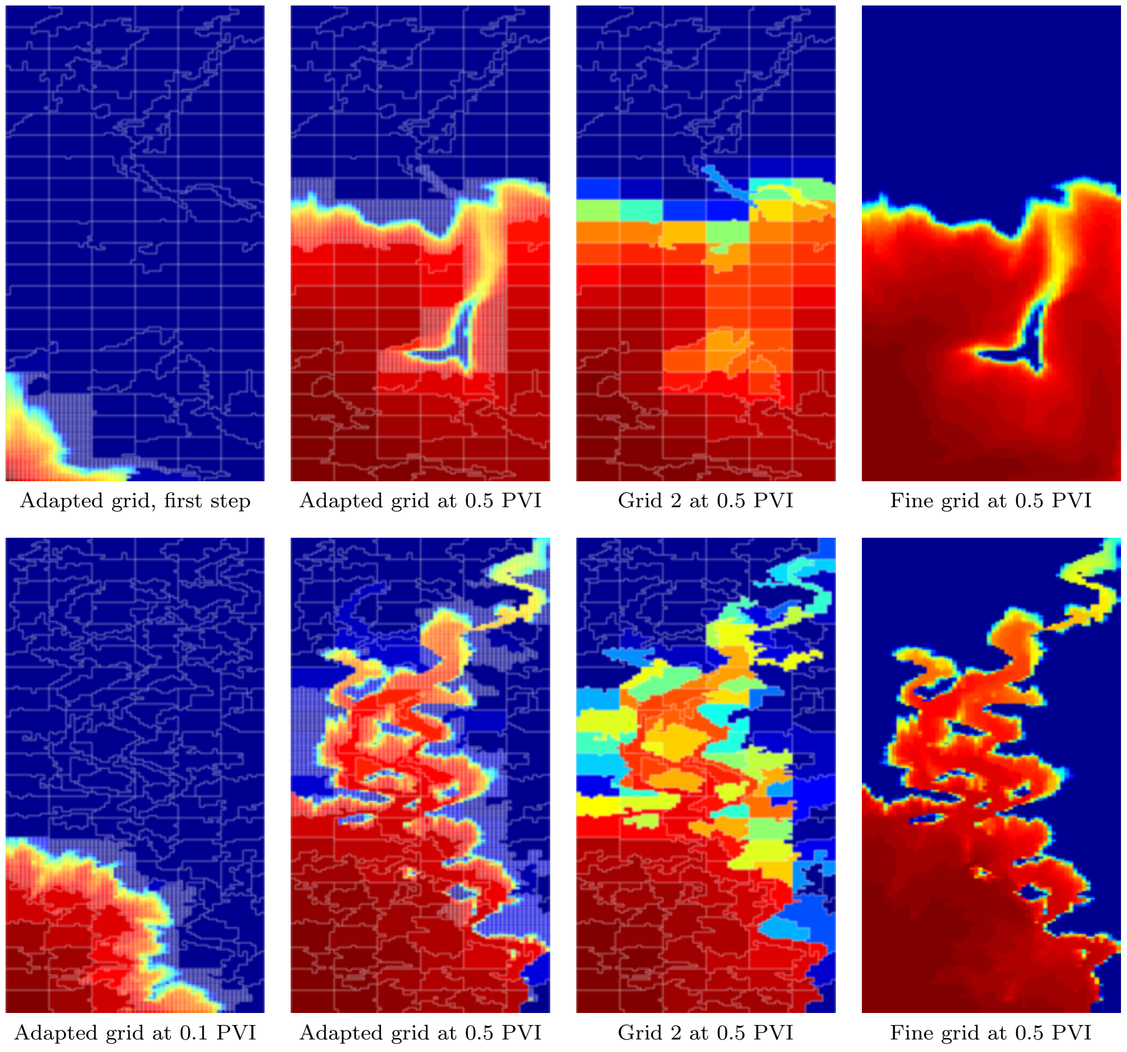


Figure 10. Examples of locally adapted grids for Layer 1 (top row) and Layer 37 (bottom row) of the SPE10 model.

Layer 37, most of the flow is confined to an intertwined pattern of narrow high-flow channels. Outside these channels, the static grid has relatively coarse blocks and once fluid enters these large low-flow blocks, the saturations get spread over a large area, causing excessive numerical diffusion.

Figure 11 shows the corresponding errors as a function of time. In the figure, we have also included results with a uniform Cartesian coarse grid with and without refinement near the water front, chosen such that the number of blocks in the Cartesian and the flow-based grids are approximately equal both with and without dynamic adaption. For both the Cartesian and the flow-based coarse grids, the coarse-scale saturation error is largest initially and decays toward water breakthrough; the qualitative behaviour of the fine-scale error is almost identical and the corresponding curves are therefore not reported. It may come as a surprise how well both the static Cartesian and the flow-based grid capture the water-cut curve for Layer 1, given the large initial error, but this result is in correspondence with previous observations

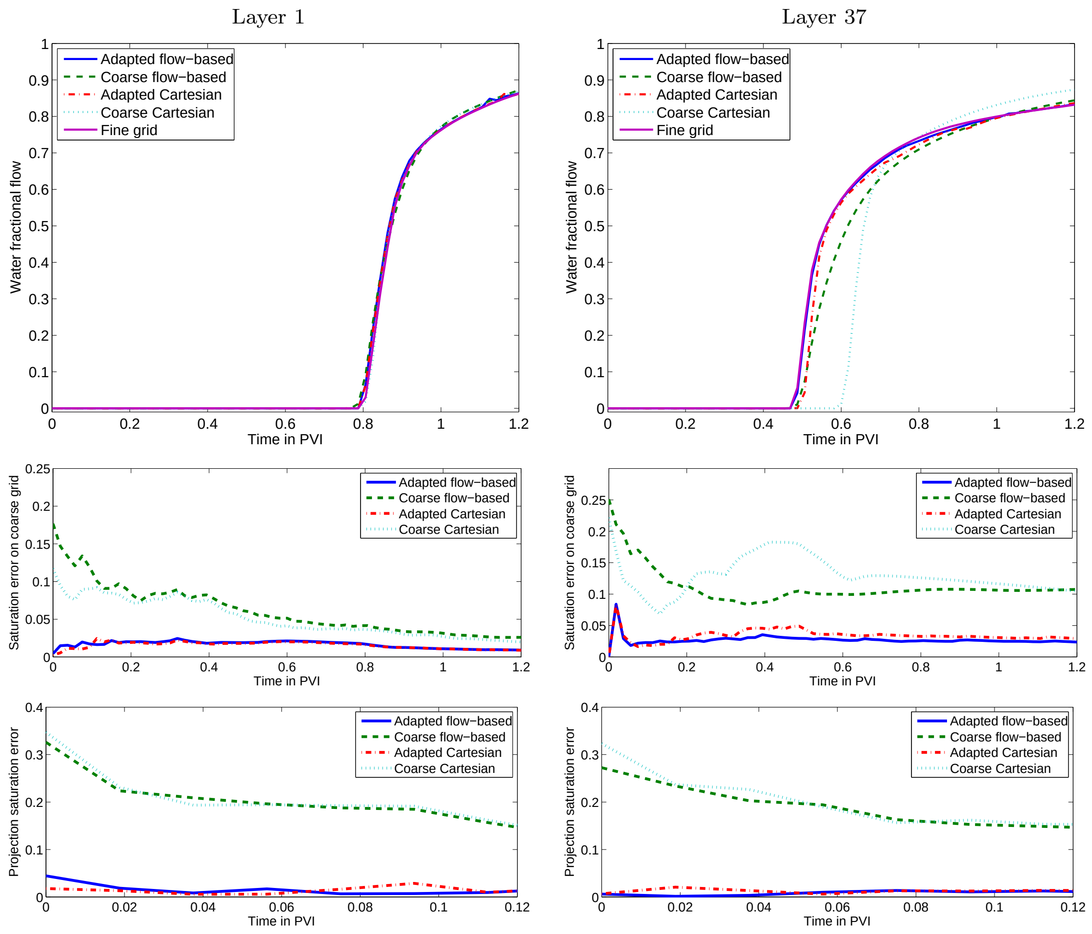


FIGURE 11. Water-cut curves and coarse-scale and projection errors as function of time for Layers 1 and 37 of the SPE10 model.

[2, 16] both for Cartesian and corner-point models. For Layer 37, we also observe how using a static flow-based grid gives a significant improvment in the water cut. The most interesting result, however, is how much both the coarse-scale error and the projection error are reduced by adding dynamical refinement. The results show that for a strong saturation front, dynamical *refinement is more important than static flow adaption. If* full *dynamic adaption is to be used, one should therefore choose the Cartesian approach, since the corresponding static partition is less expensive to generate. On the other hand, using a flow-based approach will give a better starting point for strongly heterogeneous cases. Likewise, for an unstructured fine-grid, the cost of generating a regular coarse partition is comparable to the cost associated with a flow-based partition, and the more accurate flow-based approach will therefore become attractive.*

The previous example used a simple refinement approach in which all blocks marked for refinement were replaced with the underlying fine grid. We have also experimented with more advanced options, like adding an intermediate resolution and using flow-based coarsening with finer thresholds N_L and N_U in the refinement areas. Two examples of such grids are shown in Figure 12, where we (for illustration purposes) have used a uniform Cartesian coarse grid

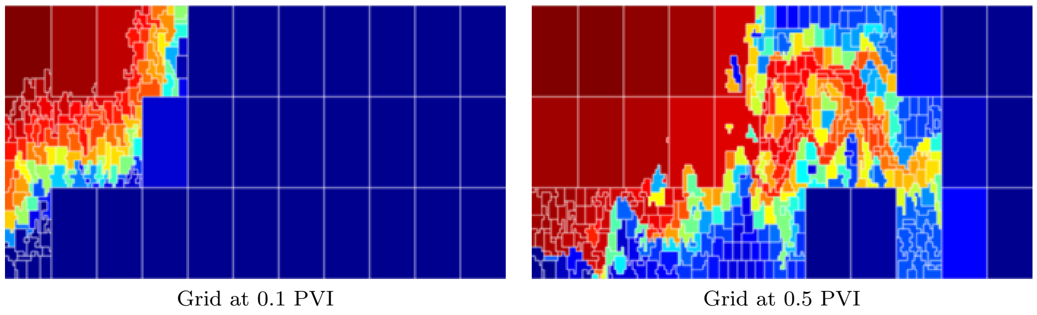


Figure 12. Uniform Cartesian coarsening with adaptive flow-based refinement for Layer 37 of SPE10. The figures show the dynamic grid at dimensionless time 0.1 and 0.5 PVI, respectively.

and added flow-based refinement dynamically along the sharp displacement fronts. Likewise, one can relatively easily implement multilevel approaches. However, for the relatively simple water-flooding scenarios we have considered, these ideas have so far not been worth the (slight) increase in algorithmic complexity.

In the previous section, we briefly discussed the need for efficient communication between a multiscale flow solver and a transport solver working on a flow-adapted grid. Introducing local refinements would potentially reintroduce the need to communicate through the fine-scale grid to dynamically provide fine-scale fluxes in refined blocks. However, it has previously been demonstrated by Kippe et al. [15] that to accurately capture the dynamic changes in the flow field, it is sufficient to update the multiscale basis functions only when a strong saturation front passes through a coarse block. Basis functions will therefore typically be updated in the regions where the transport grid is refined, and hence all the necessary fine-scale fluxes will be available. After some time steps, the strong fronts will have left the region, the coarse blocks have been reintroduced, and we can go back to use precomputed saturation mappings and sparse representation of fine-scale fluxes. For adverse mobility ratios, weak saturation fronts and smooth changes in the saturation implies that neither a dynamic refinement nor dynamic updates of basis functions are necessary.

7. Concluding Remarks

In this paper, we have shown that flow-based coarsening is a versatile method to develop efficient transport solvers that can be used in combination with multiscale flow solvers. In particular, we have started with a method proposed by Aarnes et al. [2] and shown how the partitioning computed by this method can be improved by including *a priori* information about block shapes or geology. Moreover, we have shown that using net fluxes rather than bi-directional fluxes over the coarse-block interfaces leads to both improved accuracy and computational efficiency. Finally, we have demonstrated how dynamical adaptivity easily can be included in the method to improve both the evolution and representation errors of strong saturation fronts. To simplify the presentation, most of the examples have focused on the widely-used SPE10 model. However, we have also included two examples that demonstrate that the ideas also apply to corner-point and PEBI grids. In a forthcoming paper [11], we present a general algorithmic framework for grid coarsening that includes other flow indicators based on petrophysical parameters, vorticity, and time-of-flight, as well as other neighbourhood definitions that can be used to steer the amalgamation of cells into coarse blocks.

8. Acknowledgements

The research was funded in part by Shell Norge AS and the Research Council of Norway through grants no. 175962 and 186935. The authors thank Arne Skorstad (Norwegian Computing Center) for providing access to the SAIGUP model.

9. Nomenclature

B_ℓ	=	block ℓ in the coarse grid
c_i	=	cell i in the underlying fine grid
$f(S)$	=	fractional flow function
$\Gamma_{k\ell}$	=	interface between blocks k and ℓ
γ_{ij}	=	interface between cells i and j
$h(S)$	=	fluid source/sink
$I(c), I_k$	=	flow indicator in cell c or block k
$\mathcal{N}(c)$	=	all neighbours of cell c (or for block B)
N_L, N_U	=	lower/upper bound on the number of cells in a block
n	=	number of cells in fine grid
$\mathcal{P}$	=	prolongation from coarse to fine grid
$\mathcal{R}$	=	restriction from fine to coarse grid
$\mathbf{K}, \lambda, \phi$	=	parameters: permeability, mobility, and porosity
$p, \vec{v}, S$	=	unknowns: pressure, Darcy velocity, and saturation
S_k^n	=	saturation in cell/block k at time step n
v_{ij}	=	flux between cell i and j

References

[1] J. E. Aarnes, V. Kippe, and K.-A. Lie. Mixed multiscale finite elements and streamline methods for reservoir simulation of large geomodels. *Adv. Water Resour.*, 28(3):257–271, 2005.

[2] J. E. Aarnes, V. L. Hauge, and Y. Efendiev. Coarsening of three-dimensional structured and unstructured grids for subsurface flow. *Adv. Water Resour.*, 30(11):2177–2193, 2007.

[3] J. E. Aarnes, S. Krogstad, and K.-A. Lie. Multiscale mixed/mimetic methods on corner-point grids. *Comput. Geosci.*, 12(3):297–315, 2008. doi: 10.1007/s10596-007-9072-8.

[4] M. A. Christie. Upscaling for reservoir simulation. *J. Pet. Tech.*, 48(11):1004–1010, 1996. doi: 10.2118/37324-MS.

[5] M. A. Christie and M. J. Blunt. Tenth SPE comparative solution project: A comparison of upscaling techniques. *SPE Reservoir Eval. Eng.*, 4:308–317, 2001. Url: `http://www.spe.org/csp/`.

[6] L. J. Durlofsky. Upscaling of geocellular models for reservoir flow simulation: A review of recent progress, 2003. Presented at 7th International Forum on Reservoir Simulation Bühl/Baden-Baden, Germany, June 23–27, 2003.

[7] L. J. Durlofsky. Upscaling and gridding of fine scale geological models for flow simulation, 2005. Presented at 8th International Forum on Reservoir Simulation Iles Borromees, Stresa, Italy, June 20–24, 2005.

[8] L. J. Durlofsky, R. C. Jones, and W. J. Milliken. A nonuniform coarsening approach for the scale-up of displacement processes in heterogeneous porous media. *Adv. Water Resour.*, 20:335–347, October 1997. doi: 10.1016/S0309-1708(96)00053-X.

[9] Y. Efendiev and T. Y. Hou. *Multiscale Finite Element Methods*, volume 4 of *Surveys and Tutorials in the Applied Mathematical Sciences*. Springer Verlag, 2009.

[10] C. L. Farmer. Upscaling: a review. *Int. J. Numer. Meth. Fluids*, 40(1–2):63–78, 2002. doi: 10.1002/fld.267.

[11] V. L. Hauge, K.-A. Lie, and J. R. Natvig. Grid coarsening based on amalgamation for multi-fidelity transport solvers. *submitted*, 2010.

[12] T. Y. Hou and X.-H. Wu. A multiscale finite element method for elliptic problems in composite materials and porous media. *J. Comput. Phys.*, 134:169–189, 1997.

[13] P. Jenny, S. H. Lee, and H. A. Tchelepi. Multi-scale finite-volume method for elliptic problems in subsurface flow simulation. *J. Comput. Phys.*, 187:47–67, 2003.

[14] G. Karypis and V. Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. In *International Conference on Parallel Processing*, pages 113–122, 1995.

[15] V. Kippe, J. E. Aarnes, and K.-A. Lie. A comparison of multiscale methods for elliptic problems in porous media flow. *Comput. Geosci.*, 12(3):377–398, 2008. doi: 10.1007/s10596-007-9074-6.

[16] S. Krogstad, V. L. Hauge, and A. F. Gulbransen. Adjoint multiscale mixed finite elements. *SPE J.*, SPE-119112-PA (in press; posted 23 August 2010). doi: 10.2118/119112-PA.

[17] S. Lee, H. Zhou, and H. Tchelepi. Adaptive multiscale finite-volume method for nonlinear multiphase transport in heterogeneous formations. *J. Comput. Phys*, 228(24):9036 – 9058, 2009. ISSN 0021-9991. doi: 10.1016/j.jcp.2009.09.009.

[18] T. Manzocchi et al. Sensitivity of the impact of geological uncertainty on production from faulted and unfaulted shallow-marine oil reservoirs: objectives and methods. *Petrol. Geosci.*, 14(1):3–15, 2008.

[19] J. R. Natvig and K.-A. Lie. Fast computation of multiphase flow in porous media by implicit discontinuous Galerkin schemes with optimal ordering of elements. *J. Comput. Phys.*, 227(24):10108–10124, 2008. doi: 10.1016/j.jcp.2008.08.024.

[20] J. R. Natvig and K.-A. Lie. On efficient implicit upwind schemes. In *Proceedings of ECMOR XI, Bergen, Norway, 8–11 September*. EAGE, 2008.

[21] J. R. Natvig, B. Skaflestad, F. Bratvedt, K. Bratvedt, K.-A. Lie, V. Laptev, and S. K. Khataniar. Multiscale mimetic solvers for efficient streamline simulation of fractured reservoirs. *SPE J.*, SPE-119132-PA (in press). doi: 10.2118/119132-PA.

[22] H. Zhou, S. H. Lee, and H. A. Tchelepi. Multiscale finite volume formulation for the saturation equations. In *SPE Reservoir Simulation Symposium, The Woodlands, TX, USA, 2–4 February 2009*, 2009. doi: 10.2118/119183-MS.