15
Upscaling Petrophysical Properties

In Chapter 2 we argued that porous rock formations are heterogeneous across a wide span of length scales, from the micrometer scale of pore channels to the kilometer scale of petroleum reservoirs or even larger scales for aquifer systems and sedimentary basins. Describing all flow processes pertinent to hydrocarbon recovery or CO_2 storage with a single model is therefore not possible. Geological characterization therefore usually involves a hierarchy of models that each covers a limited range of physical scales cover a wide range of physical scales. Section 2.3 introduced you to the concept of representative elementary volumes and mentioned various types of models used for flow studies including flow in core samples (cm scale), bed models (meter scale), sector models (tens to hundreds of meters), and field models (km scale). Geocellular models at the field/sector scale are made to represent the heterogeneity of the reservoir and possibly incorporate a measure of inherent uncertainty in reservoir simulation. Pore, core, and bed models are mainly designed to give input to the geological characterization and to derive flow parameters for simulation models. All model types must be calibrated against static and dynamic data of very different spatial (and temporal) resolution: thin sections, core samples, well logs, geological outcrops, seismic surveys, well tests, production data, core flooding and other laboratory experiments, etc. To learn more about the process of building reservoir models, you should consult the excellent textbook by Ringrose and Bentley [265].

To systematically propagate the effects of small-scale geological variations observed in core samples up to the reservoir scale, we need mathematical methods that can *homogenize* a detailed fine-scale model and replace it with an equivalent model on a coarser scale that contains much fewer parameters that represent the behavior of the fine-scale model in an averaged sense; see Figure 15.1.

Such techniques are not only used to incorporate the effect of small-scale models into macroscale models. High-resolution geocellular models used for reservoir characterization on field or sector scale tend to contain more volumetric cells than contemporary simulators can handle without having to resort to massively parallel high-performance computing. As a result, flow simulations are usually performed with models containing less details than those used for characterization. Even if you had all the computational power you needed, there are several arguments why you should still perform your simulations on coarser models. First of all, one can argue that high-resolution models tend to contain more

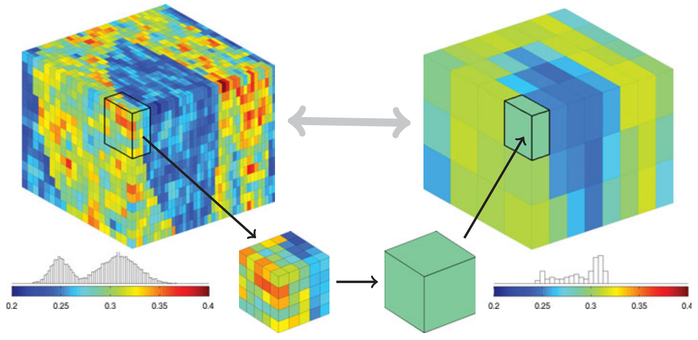


Figure 15.1 Upscaling of petrophysical properties, here represented by porosity.

details than what is justified by the information in the available data. Second, in many modeling workflows the available computational power should be spent running multiple model realizations to span the range of plausible outcomes rather than on obtaining high numerical resolution for a few highly uncertain predictions. Third, because coarser models contain fewer parameters, they are simpler to calibrate to observed reservoir responses like pressure tests and production data. Finally, a coarser model may be sufficient to predict flow patterns and reservoir responses with the accuracy needed to make a certain business decision. It is perhaps tempting to believe that with future increases in computing power, one will soon be able to close the gap in resolution between models used for reservoir characterization and models used for flow simulation. The development so far indicates that such an idea is wrong. The trend is rather that increases in computational power enable geologists and reservoir engineers to build larger and more complex geocellular models at a pace that outperforms the improvement in simulation capabilities.

For all these reasons, there is a strong need for mathematical and numerical techniques that can be used to communicate parameters and properties between models of different spatial resolution. This need will persist, and maybe even grow stronger, in the foreseeable future. This chapter discusses techniques for upscaling static rock properties.

15.1 Upscaling for Reservoir Simulation

Upscaling (or homogenization) refers to the process of propagating properties and parameters from a model of a given resolution to a model of *lower spatial resolution*. In this process, *heterogeneous regions* in a reservoir model are replaced by *homogeneous regions* to make up a coarser model of the same reservoir. Or, in the words of Chapter 14 on grid coarsening, upscaling is the process in which petrophysical properties in the cells that make up a coarse block are averaged into effective values for each coarse block; see Figure 15.1. The effective properties of the new homogeneous regions are defined so that they preserve the effects of small-scale variations in an averaged sense. How this averaging should be performed depends on the type of property to be upscaled. We distinguish between *additive properties* that can be upscaled using simple volumetric averaging and *nonadditive* *properties*, for which correct averaging methods only exist in special cases and the best you can hope for in the general case is to compute accurate approximations.

Downscaling refers to the process of propagating properties from a model with a given spatial resolution to a model having *higher spatial resolution*. We can, for instance, be interested in refining coarse-scale modifications in petrophysical properties obtained during a history match on a simulation model to update the underlying geomodel. In the downscaling process, the aim is to preserve both the coarse-scale trends and the fine-scale heterogeneity structures.

The process of upscaling petrophysical parameters leads to many fundamental questions. For instance, do the partial differential equations that make up the flow model on the coarse scale take the same form as the equations modeling flow at the subgrid scale? And if so, how do we honor the fine-scale heterogeneities at the subgrid level? Even though upscaling has been a standard procedure in reservoir simulation for more than four decades, nobody has fully answered these questions rigorously, except for cases with special heterogeneous formations such as periodic or stratified media.

Homogenization is a rigorous mathematical theory for asymptotic analysis of periodic structures; see [41, 149, 135]. A relevant result states that for a periodic medium with permeability $\mathbf{K}(\frac{x}{\varepsilon})$, there exists a constant symmetric and positive-definite tensor $\mathbf{K}_0$ such that the solutions p_ε and $v_\varepsilon = -\mathbf{K}(\frac{x}{\varepsilon})\nabla p_\varepsilon$ to the elliptic problem

$$-\nabla \cdot \mathbf{K}(\frac{x}{\varepsilon})\nabla p_\varepsilon = q \tag{15.1}$$

converge uniformly as $\varepsilon \to 0$ to the solution of the homogenized equation

$$-\nabla \cdot \mathbf{K}_0 \nabla p_0 = q. \tag{15.2}$$

Homogenization theory can be used to derive homogenized tensors for the region, if we can safely assume that the region we seek to compute an effective property for is part of an infinite periodic medium. The main advantage of homogenization is that it provides mathematical methods to prove existence and uniqueness of the coarse-scale solution and also verifies that the governing equation at the macroscopic level takes the same form as the elliptic equation that governs porous media flow at the level of the representative elementary volumes (REVs). However, it is more debatable whether mathematical homogenization can be used for practical simulations, since natural rocks are rarely periodic.

As you will see shortly, the most basic upscaling techniques rely on local averaging procedures that calculate effective properties in each grid block solely from properties within the grid block. Because these averaging procedures do not consider coupling beyond the local domain, they fail to account for the effect of long-range correlations and large-scale flow patterns in the reservoir, unless their effect can be represented correctly by the forces that drive flow inside the local domain. Different flow patterns may call for different upscaling procedures, and it is generally acknowledged that global effects must also be taken into consideration to obtain robust coarse-scale simulation models.

Upscaling must also be seen in close connection with griding and coarsening methods, as discussed in Chapters 3 and 14. To make a fine-scale model, you must decide what kind of grid you want to represent the porous medium, what (local) resolution you need, and how you should orient your grid cells. Obviously, you should consider the same questions for the upscaled model. To make an upscaled simulation model as accurate and robust as possible, the grid should be designed so that grid blocks capture heterogeneities on the scale of the block. This often implies that you may need significantly more blocks if you use a regular grid than if you use an unstructured polyhedral grid. You may also want to use different coarsening factors in zones of high and low flow, near and far away from wells and fluid contacts, as discussed in Chapter 14. The importance of designing a good coarse grid should not be underestimated. Henceforth, however, we simply assume that a suitable coarse grid is available and focus on upscaling techniques.

The literature on upscaling techniques is extensive, ranging from simple averaging techniques, e.g., [151], via flow-based methods that rely on local flow problems [37, 90] to more comprehensive global [225, 133] and local–global [63, 64, 117] methods. Some attempts have been made to analyze the upscaling process, e.g., [31, 318], but so far there is generally no theory or framework for assessing the quality of an upscaling technique. In fact, upscaling techniques are rarely rigorously quantified with mathematical error estimates. Instead, the quality of upscaling techniques is usually assessed by comparing upscaled production characteristics with those obtained from a reference solution computed on an underlying fine grid. This deficiency is one of the motivations behind the development of modern multiscale methods [95, 195]. Such multiscale methods are more well developed in MRST than in any comparable software, but a detailed discussion is outside the scope of this book.

In the following, we only discuss the basic principles and try to show you how you can implement relatively simple upscaling methods; we also attempt to explain why some methods may work well for certain flow scenarios and not for others. If you are interested in a comprehensive overview, you should consult one of the many review papers devoted to this topic, e.g., [70, 311, 31, 263, 108, 91, 92]. We start by a brief discussion of how to upscale porosity and other additive properties, before we move on to discuss upscaling permeability, which is the primary example of a nonadditive property. Because of the way Darcy's law has been extended from single-phase to multiphase flow, it is common to distinguish the upscaling of absolute permeability $\mathbf{K}$ from the upscaling of relative permeability $k_{r\alpha}$. Upscaling of absolute permeability is often called *single-phase upscaling*, whereas upscaling of relative permeability is referred to as *multiphase upscaling*. The main parts of this chapter are devoted to permeability upscaling and to upscaling of the corresponding transmissibilities that account for permeability effects in finite-volume discretizations. As in the rest of the book, our discussion will to a large extent be driven by examples, for which you can find complete codes in the `upscaling` directory of the `book` module.

15.2 Upscaling Additive Properties

Porosity is the simplest example of an *additive property* and can be upscaled through a simple volumetric average. If Ω denotes the region we want to average over, the averaged porosity value is given as

$$\phi^* = \frac{1}{\Omega}\int_\Omega \phi(\vec{x})\,d\vec{x}. \tag{15.3}$$

Implementing the computation of this volumetric average can be a bit tricky if the coarse blocks and the fine cells are not matching. In MRST, however, we always assume that the coarse grid is given as a partition of the fine grid, as explained in Chapter 14. If `q` denotes the vector of integers describing the coarse partition, upscaling porosity amounts to a single statement

```
crock.poro = accumarray(q,rock.poro.*G.cells.volumes)./ ...
             max(accumarray(q,G.cells.volumes),eps);
```

We use `max(..,eps)` to safeguard against division by zero in case your grid contains blocks with zero volume or your partition vector is not contiguous. Weighting by volume is not necessary for grids with uniform cell sizes. Similar statements were used to go from the fine-scale model on the left side of Figure 15.1 to the coarse-scale model shown on the right. Complete source code is found in the script `illustrateUpscaling.m`.

Other additive (or volumetric) properties like net-to-gross, (residual) saturations, and concentrations can be upscaled almost in the same way, except that you should replace the bulk average in (15.3) by a *weighted* average. If n denotes net-to-gross, the correct average would be to weight with porosity, so that

$$n^* = \Big[\int_\Omega \phi(\vec{x})\,d\vec{x}\Big]^{-1}\int_\Omega \phi(\vec{x})n(\vec{x})\,d\vec{x}. \tag{15.4}$$

and likewise for saturations. In MRST, we can compute this upscaling as follows:

```
pv = rock.poro.*G.cells.volumes;
N  = accumarray(q,pv.*n)./ max(accumarray(q,pv),eps);
```

To verify that this is the correct average, we simply compute

$$\begin{aligned}\phi^* n^* &= \Big[\frac{1}{\Omega}\int_\Omega \phi(\vec{x})\,d\vec{x}\Big]\Big[\int_\Omega \phi(\vec{x})\,d\vec{x}\Big]^{-1}\int_\Omega \phi(\vec{x})n(\vec{x})\,d\vec{x}\\ &= \frac{1}{\Omega}\int_\Omega \phi(\vec{x})\,n(\vec{x})\,d\vec{x} = (\phi\,n)^*.\end{aligned}$$

Using the same argument, you can easily propose that concentrations should be weighted with saturations, and so on. Rock type (or flow unit), on the other hand, is not an additive property, even though it is sometimes treated almost as if it was by applying a majority vote to identify the rock type that occupies the largest volume fraction of a block. Such a simple approach is not robust [324] and should generally be avoided.

To get more acquainted with upscaling of additive quantities, I recommend that you try to do the following computer exercises.

COMPUTER EXERCISES

15.2.1 Construct a $8 \times 24 \times 5$ coarse grid `CG` of the SAIGUP model and upscale the additive rock properties. Verify your upscaling by computing

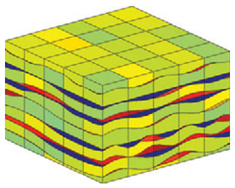
```
pv = accumarray(CG.partition, poreVolume(G,rock));
max(abs(pv - poreVolume(CG, crock)))
```

Plot histograms of the fine and coarse porosities and compare. How would you make a coarse grid that preserves the span in porosity values better? To measure the quality of a given upscaling, you can for instance use the following error measure:

```
err = sum(abs(rock.poro - crock3.poro(CG.partition)))/...
        sum(rock.poro)*100;
```

It may also be helpful to plot the cell-wise discrepancy between the upscaled and the original porosity over the fine grid.

15.2.2 The file `mortarTestModel` from the `BedModels1` data set describes a sedimentary bed consisting of three different facies. Construct a coarse model that has 5×5 blocks in the lateral direction and as many blocks in the vertical direction as required to preserve distinct facies layers, as shown in the plot to the right. Upscale the porosity of the model. Do you think the resulting coarse grid is suitable for flow simulations? (Hint: Check the guidelines from Section 14.7.)



15.3 Upscaling Absolute Permeability

To study upscaling of absolute permeability, it is sufficient to consider single-phase flow in the form of a variable-coefficient Poisson equation

$$\nabla \cdot \mathbf{K} \nabla p = 0. \tag{15.5}$$

Even with such a simple equation, the choice of what is the best method to average absolute permeability generally depends on a complex interplay between the local permeability distribution and the characteristic flow directions. In certain special cases, one can develop simple methods that average permeability correctly, but in the general case, all you can do is develop computational methods that approximate the true effective permeability of the upscaled region. How accurate a given approximation is will depend on the coarse grid, the specific upscaling method, the purpose for which the upscaled values are to be used, and the complexity of the fine-scale permeability distribution.

Most techniques for upscaling absolute permeability seek an averaged tensor $\mathbf{K}^*$ that reproduces the same total flow through each homogeneous region you would obtain by solving the single-pressure equation (15.5) with the full fine-scale heterogeneity. In other words, if Ω is the homogeneous region to which we wish to assign an effective property $\mathbf{K}^*$, this property should fulfill

$$\int_\Omega \mathbf{K}(\vec{x})\nabla p \, d\vec{x} = \mathbf{K}^* \int_\Omega \nabla p \, d\vec{x}. \tag{15.6}$$

This equation states that the net flow rate $\vec{v}_\Omega$ through Ω is related to the average pressure gradient $\nabla_\Omega p$ in Ω through the upscaled Darcy law

$$\vec{v}_\Omega = -\mathbf{K}^* \nabla_\Omega p. \tag{15.7}$$

The upscaled permeability tensor $\mathbf{K}^*$ is not uniquely defined by (15.6) for a given pressure field p, and conversely, there does not exist a unique $\mathbf{K}^*$ so that (15.6) holds for any pressure field. This reflects that $\mathbf{K}^*$ depends on the flow through Ω, which in turn is determined by the boundary conditions specified on $\partial\Omega$. The better you know the boundary conditions that a homogenized region will be subject to in subsequent simulations, the more accurate estimates you can compute for the upscaled tensor $\mathbf{K}^*$. In fact, if you know these boundary conditions exactly, you can compute the true effective permeability. In general, you will not know these boundary conditions in advance unless you have already solved your problem, and the best you can do is to make an educated and representative guess that aims to give reasonably accurate results for a specific or a wide range of flow scenarios. Another problem is that even though the permeability tensor of a physical system must be symmetric and positive definite (i.e., $\vec{z} \cdot \mathbf{K}\vec{z} > 0$ for all nonzero $\vec{z}$), there is generally no guarantee that the effective permeability tensor constructed by an upscaling algorithm fulfills the same properties. The possible absence of symmetry and positive definiteness shows that the single-phase upscaling problem is fundamentally ill-posed.

15.3.1 Averaging Methods

The simplest way to upscale permeability is to use an explicit averaging formula. *power averages* constitute a general class of such formulas,

$$\mathbf{K}^* = A_p(\mathbf{K}) = \Big(\frac{1}{|\Omega|}\int_\Omega \mathbf{K}(\vec{x})^p \, d\vec{x}\Big)^{1/p}. \tag{15.8}$$

Here, $p = 1$ and $p = -1$ correspond to the arithmetic and harmonic average, respectively, whereas the geometric mean is obtained in the limit $p \to 0$ as

$$\mathbf{K}^* = A_0(\mathbf{K}) = \exp\Big(\frac{1}{|\Omega|}\int_\Omega \log(\mathbf{K}(\vec{x}))\, d\vec{x}\Big). \tag{15.9}$$

The use of power averaging can be motivated by the so-called Wiener-bounds [317], which state that for a statistically homogeneous medium, the correct upscaled permeability will be bounded above and below by the arithmetic and harmonic mean, respectively.

To motivate (15.8), we can look at the relatively simple problem of upscaling permeability within a one-dimensional domain $[0, L]$. From the flow equation (15.5) and Darcy's law, we have that

$$-\big(K(x)p'(x)\big)' = 0 \quad \Longrightarrow \quad v(x) = K(x)p'(x) \equiv \text{constant.}$$

The upscaled permeability K^* must satisfy Darcy's law on the coarse scale, hence $v = -K^*[p_L - p_0]/L$. Alternatively, this formula can be derived from (15.6)

$$K^* \int_0^L p'(x)dx = K^*(p_L - p_0) \overset{(15.6)}{=} \int_0^L K(x)p'(x)dx = -\int_0^L v dx = -Lv.$$

We can now use (15.6) and Darcy's law to find the expression for K^*

$$\begin{aligned}\int_0^L p'(x)dx &= -\int_0^L \frac{v}{K(x)}dx \\ &= K^*\frac{p_L - p_0}{L}\int_0^L \frac{1}{K(x)}dx = K^*\Big(\int_0^L p'(x)dx\Big)\Big(\frac{1}{L}\int_0^L \frac{1}{K(x)}dx\Big),\end{aligned}$$

from which it follows that the correct way to upscale K is to use the harmonic average

$$K^* = \Big(\frac{1}{L}\int_0^L \frac{1}{K(x)}dx\Big)^{-1}. \tag{15.10}$$

This result is universally valid only in one dimension, but also applies to the special case of a perfectly stratified isotropic medium with layers perpendicular to the direction of pressure drop as illustrated in Figure 15.2. It is a straightforward exercise to extend the analysis just presented to prove that harmonic averaging is the correct upscaling in this case. (If you are familiar with Ohm's law in electricity, you probably realize that this setup is similar to that of resistors set in parallel.)

Computing the harmonic average is straightforward in MRST:

```
vol  = G.cells.volumes;
for i=1:size(rock.perm,2)
   crock.perm(:,i) = accumarray(q,vol) ./ ...
                     accumarray(q,vol./rock.perm(:,i))
end
```

Here, we have assumed as in the previous section that q is a partition vector describing the coarse grid.

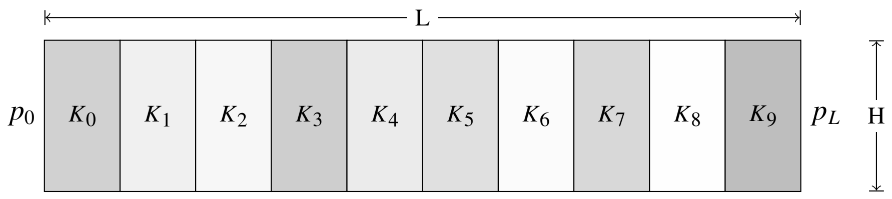


Figure 15.2 Example of a perfectly stratified isotropic medium with layers perpendicular to the direction of the pressure drop, for which harmonic averaging is the correct way to upscale absolute permeability.

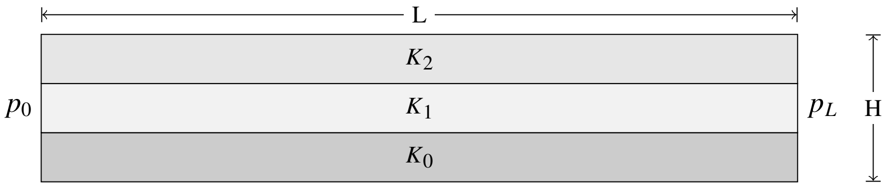


Figure 15.3 Example of a perfectly stratified isotropic medium with layers parallel to the direction of the pressure drop, for which arithmetic averaging is the correct way to upscale absolute permeability.

Another special case is when the layers of a perfectly stratified isotropic medium are parallel to the pressure drop as shown in Figure 15.3. Because the permeability is constant along the direction of pressure drop, the pressure will be linear in the x-direction with $p(x, y) = p_0 + x(p_L - p_0)/L$. Using (15.6), we can compute

$$\begin{aligned} K^* \int_0^H \int_0^L \partial_x p(x, y)\, dx\, dy &= K^* H(p_L - p_0) \\ &= \int_0^H \int_0^L K(x, y)\partial_x p(x, y)\, dx\, dy = \frac{p_L - p_0}{L} \int_0^H \int_0^L K(x, y)\, dx\, dy, \end{aligned}$$

from which it follows that the correct upscaling is to use the arithmetic average

$$K^* = \frac{1}{LH} \int_0^H \int_0^L K(x, y)\, dx\, dy. \tag{15.11}$$

These examples show that averaging techniques can give correct upscaling in special cases, also in three dimensional space. If we now combine these two examples, we see that we can define the following upscaled permeability tensors for the isotropic media shown in Figures 15.2 and 15.3,

$$\mathbf{K}^* = \begin{bmatrix} A^x_{-1}(K) & 0 \\ 0 & A^x_1(K) \end{bmatrix} \quad \text{and} \quad \mathbf{K}^* = \begin{bmatrix} A^y_1(K) & 0 \\ 0 & A^y_{-1}(K) \end{bmatrix},$$

where the superscripts x and y on the averaging operator A from (15.8) signify that the operator is only applied in the corresponding spatial direction. These averaged permeabilities would produce the correct net flow across the domain when these models are subject to a pressure differential between the left and right boundaries or between the top and bottom boundaries. For other boundary conditions, however, the upscaled permeabilities generally give incorrect flow rates.

To represent flow in more than one direction, also for cases with less idealized heterogeneous structures modeled by a diagonal fine-scale tensor, you can compute a permeability tensor with the following diagonal components:

$$\mathbf{K}^* = \begin{bmatrix} A^{yz}_1(A^x_{-1}(\mathbf{K})) & 0 & 0 \\ 0 & A^{xz}_1(A^y_{-1}(\mathbf{K})) & 0 \\ 0 & 0 & A^{xy}_1(A^z_{-1}(\mathbf{K})) \end{bmatrix}.$$

In other words, we start by computing the harmonic average of each of the diagonal permeability components in the corresponding longitudinal direction, i.e., compute the harmonic average of K_{xx} along the x direction, and so on. Then, we compute the arithmetic average in the transverse directions, i.e., in the y and z directions for K_{xx}, and so on. This average is sometimes called the *harmonic-arithmetic average* and may give reasonable upscaling for layered reservoirs when the primary direction of flow is along the layers. More importantly, the harmonic-arithmetic average provides a tight lower bound on the effective permeability, whereas the opposite method, the arithmetic-harmonic average, provides a tight upper bound.

It is not obvious how to compute the harmonic-arithmetic average for a general unstructured grid, but it is almost straightforward to do it for rectilinear and corner-point grids in MRST. For brevity, we only show the details for the case when the grid has been partitioned uniformly in index space:

```
q = partitionUI(G, coarse);
vol = G.cells.volumes;
for i=1:size(rock.perm,2)
   dims = G.cartDims; dims(i)=coarse(i);
   qq = partitionUI(G, dims);
   K = accumarray(qq,vol)./accumarray(qq,vol./rock.perm(:,i));
   crock.perm(:,i) = accumarray(q,K(qq).*vol)./accumarray(q,vol);
end
```

The key idea in the implementation above is that to compute the harmonic average in one axial direction, we introduce a temporary partition `qq` that coincides with the coarse grid along the given axial direction and with the original fine grid in the other axial directions. This way, the call to `accumarray` has the effect that the harmonic average is computed for one longitudinal stack of cells at the time inside each coarse block. To compute the arithmetic average, we simply map the averaged values back onto the fine grid and use `accumarray` over the original partition `q`.

Example 15.3.1 *The script* `averagingExample1` *in the* `book` *module shows an example of the averaging methods just discussed. The permeability field is a 40 × 60 sub-sample of Layer 46 from the fluvial Upper Ness formation in the SPE 10 data set. Figure 15.4 compares the effective permeabilities on a 15 × 15 coarse grid computed by arithmetic, harmonic, and harmonic-arithmetic averaging. We see that arithmetic averaging has a tendency to preserve high permeability values, harmonic averaging tends to preserve small permeabilities, whereas harmonic-arithmetic averaging is somewhere in between.*

Although simple averaging techniques can give correct upscaling in special cases, they tend to perform poorly in practice because the averages do not properly reflect the heterogeneity structures. Likewise, it is also generally difficult to determine which averaging to use, since the best averaging method depends both on the heterogeneity of the reservoir and the prevailing flow directions. Let us illustrate this by an example.

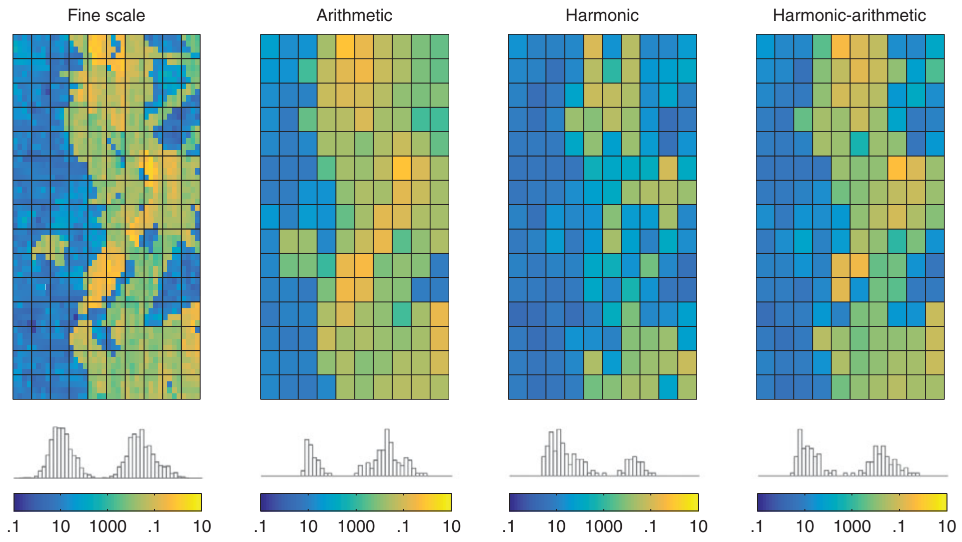


Figure 15.4 Arithmetic, harmonic, and harmonic-arithmetic averaging applied to a 40 × 60 subset of Layer 46 from the SPE 10 data set.

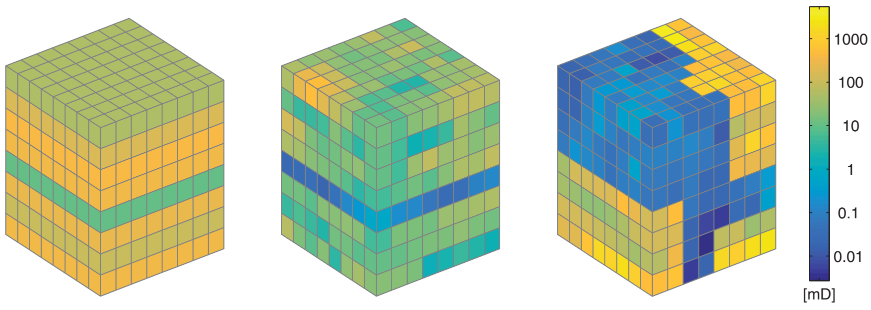


Figure 15.5 Three models used to test the accuracy of averaging techniques. From left to right: a layered model and two subsets from the Tarbert and the Upper Ness formation, respectively, from the SPE 10 data set.

Example 15.3.2 *Consider an 8 × 8 × 8 reservoir with three different permeability realizations shown in Figure 15.5, which we upscale to a single coarse block using arithmetic, harmonic, and harmonic-arithmetic averaging. To assess the quality of the upscaling, we compare fine-scale and coarse-scale prediction of net flux across the outflow boundary for three different flow patterns: from west to east, from south to north, and from bottom to top. Complete source code for this simulation setup is given in the script* `averagingExample2`.

Table 15.1 reports the ratio between the outflow computed by the coarse models and the outflow computed on the original fine grid. For the layered model, arithmetic and harmonic-arithmetic averaging correctly reproduces flow in the lateral directions, whereas

Table 15.1 *Ratio between flow rate predicted by upscaled/fine-scale models for three permeability fields, flow scenarios, and upscaling methods.*

Model	Flow pattern	Arithmetic	Harmonic	Harm-arith
Layered	East→West	1.0000	0.2662	1.0000
	North→South	1.0000	0.2662	1.0000
	Top→Bottom	3.7573	1.0000	1.0000
Tarbert	East→West	1.6591	0.0246	0.8520
	North→South	1.6337	0.0243	0.8525
	Top→Bottom	47428.0684	0.3239	0.8588
Upper Ness	East→West	3.4060	0.0009	0.8303
	North→South	1.9537	0.0005	0.7128
	Top→Bottom	6776.8493	0.0020	0.3400

flow normal to the layers is correctly reproduced by harmonic and harmonic-arithmetic averaging. For the two anisotropic models from SPE 10, on the other hand, the flow rates predicted by the arithmetic and harmonic methods are generally far off. The combined harmonic-arithmetic method is more accurate, with less than 15% discrepancy for the Tarbert model and 17–76% discrepancy for the Upper Ness model. Whether this can be considered an acceptable result will depend on what purpose the simulation is to be used for.

COMPUTER EXERCISES

15.3.1 Set up a set of flow simulations with different boundary conditions and/or well patterns to test the accuracy of the three effective permeability fields computed in `averagingExample1`.

15.3.2 Implement harmonic-arithmetic averaging for the SAIGUP model. (Hint: the script `cpGridHarmonic` in the `upscaling` module computes harmonic averaging for the SAIGUP model.) Can you also apply it to the coarse grids generated for CaseB4 in the previous chapter?

15.3.3 Redo the upscaling in Figure 15.4 using a coarse grid that adapts to high-permeable or high-flow zones. Does this help preserve the distribution of the permeabilities?

15.3.4 Implement the operator A_p in (15.8) as a new utility function.

15.3.5 How would you upscale the well indices in a Peaceman well model?

15.3.2 Flow-Based Upscaling

Thus far in this section, we repeatedly used simple flow problems to argue whether a particular averaging method was good or not. Taking this idea one step further, we could impose representative boundary conditions along the perimeter of each coarse block and solve

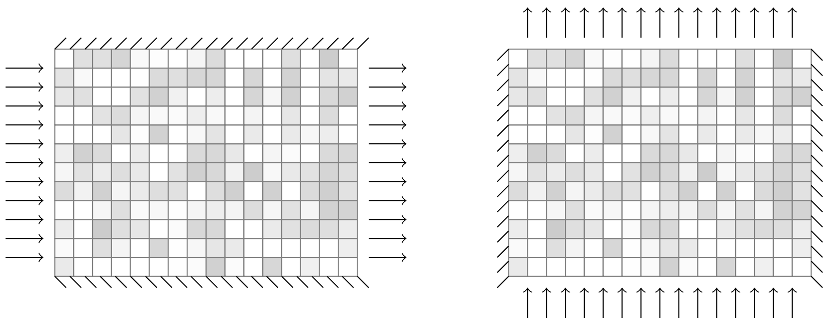


Figure 15.6 Illustration of a simple flow-based upscaling method, solving $-\nabla \cdot (\mathbf{K}\nabla p) = 0$, with $p = 1$ and $p = 0$ prescribed along the inflow and outflow boundaries respectively, and no-flow boundary conditions elsewhere.

the flow problem (15.5) numerically to determine fine-scale pressures and flow rates and corresponding net flow rates and average pressure gradients. We could then invert the flow equations discretized on the coarse block to determine effective coarse-scale permeabilities from Darcy's law. That is, once p and v have been computed, we use the same type of argument as we used to determine the formulas for the harmonic and arithmetic averages, except that we now work with discrete pressures and fluxes. This raises the immediate question of what kind of boundary conditions we should use.

Different boundary conditions have been suggested over the years. One obvious approach is to use three different boundary conditions for each block to create a pressure drop in each axial directions in the same way as for the computations reported in Table 15.1; see Figure 15.6. By specifying sealing boundary conditions along the other boundaries, we ensure that the effective flow over the block follows the same axial direction as the pressure drop. This setup emulates how permeability is measured on core samples in the laboratory and provides us with three pairs of flow rates in 3D that we can use to compute an upscaled permeability tensor with diagonal elements

$$K_{xx} = -\frac{v_x L_x}{\Delta p_x}, \qquad K_{yy} = -\frac{v_y L_y}{\Delta p_y}, \qquad K_{zz} = -\frac{v_z L_z}{\Delta p_z}.$$

Here, v_x is net flux, L_x the characteristic length of the block, and Δp_x the pressure drop inside the block in the x-direction. With this method, the off-diagonal elements are zero by construction. Strictly speaking, the assumption of sealing boundaries is only valid in the idealized case when the permeability field is symmetric with respect to the faces of the coarse grid, i.e., if the coarse block is surrounded by mirror images of itself. It can be shown that this method tends to introduce an upscaling bias towards low permeability values by thickening shale barriers and narrowing sand channels [163, 161].

Unless the grid block is located next to a sealing fault or an impermeable layer, it therefore more natural to assume that the block has open boundaries so that a pressure differential applied along one of the axial directions will induce a net flow also in transverse directions. This means that the effective permeability is a full tensor. One method to emulate open boundaries is to prescribe a constant pressure on faces perpendicular to each flow direction and linear pressure drop along the sides that are parallel to the flow direction [120, 266] so that the flow can leave or enter these sides. As for the sealing boundary, a unit pressure drop is applied in each of the axial directions. Use of linear boundary conditions is only strictly valid if the heterogeneous coarse block is embedded inside a homogeneous medium and tends to produce permeabilities with a bias towards high values [90].

Another popular option is to prescribe *periodic boundary conditions* [90], assuming that the grid block is sampled from a periodic medium so that the fluxes in and out of opposite boundaries are equal. In other words, to compute the x-component, we impose the following conditions:

$$\begin{aligned} p(L_x,y,z) &= p(0,y,z) - \Delta p, & v(L_x,y,z) &= v(0,y,z), \\ p(x,L_y,z) &= p(x,0,z), & v(x,L_y,z) &= v(x,0,z), \\ p(x,y,L_z) &= p(x,y,0), & v(x,y,L_z) &= v(x,y,0), \end{aligned} \tag{15.12}$$

and similarly for the other axial directions. This approach gives a symmetric and positive-definite tensor and is usually more robust than specifying sealing boundaries. Periodic boundaries tend to give permeabilities that lie in between the lower and upper bounds computed using sealing and linear boundaries, respectively.

Let us see how two of these methods can be implemented in MRST for a rectilinear or corner-point grid. For simplicity, we assume that the `d`-dimensional domain to be upscaled is rectangular and represented by grid `G` and rock structure `rock`. We start by setting up structures representing boundary conditions,

```
bcsides = {'XMin', 'XMax'; 'YMin', 'YMax'; 'ZMin', 'ZMax'};
for j = 1:d;
   bcl{j} = pside([], G, bcsides{j, 1}, 0);
   bcr{j} = pside([], G, bcsides{j, 2}, 0);
end
Dp = {4*barsa, 0};
L  = max(G.faces.centroids)-min(G.faces.centroids);
```

The first structure, `bcsides`, just contains name tags to locale the correct pair of boundary faces for each flow problem. The `bcl` and `bcr` structures are used to store a template of all the boundary conditions we must use. This is not needed to implement the pressure-drop method, but these structures are handy when setting up periodic boundary conditions. Finally, `Dp` and `L` contain the prescribed pressure drop and the characteristic length in each axial direction. With the data structures in place, the loop that does the upscaling for the pressure-drop method is quite simple:

```
for i=1:d
  bc = addBC([], bcl{i}.face, 'pressure', Dp{1});
  bc = addBC(bc, bcr{i}.face, 'pressure', Dp{2});
  xr = incompTPFA(initResSol(G, 100*barsa, 1),< G, hT, fluid, 'bc', bc);
  v(i)  = sum(xr.flux(bcr{i}.face)) / sum(G.faces.areas(bcr{i}.face));
  dp(i) = Dp{1}/L(i);
end
K = convertTo(v./dp, milli*darcy);
```

That is, we loop over the axial directions and for each direction we: (i) specify a pressure from left to right, (ii) compute pressures and fluxes by solving the resulting Poisson problem, and (iii) compute average velocity `v` across the outflow boundary and average pressure drop `d`. Finally, we can compute the effective permeability by inverting Darcy's law. The implementation just described is a key algorithmic component in steady-state upscaling of relative permeabilities for multiphase flow and is therefore offered as a separate utility function `upscalePermeabilityFixed` in the `upscaling` module.

Periodic boundary conditions are slightly more involved. We start by calling a routine that modifies the grid structure so that it represents a periodic domain:

```
[Gp, bcp] = makePeriodicGridMulti3d(G, bcl, bcr, Dp);
for j=1:d, ofaces{j} = bcp.face(bcp.tags==j); end
```

Technically, the grid is extended with a set of additional faces that connect cells on opposite boundaries of the domain so that the topology becomes toroidal. The routine also sets up an appropriate structure for representing periodic boundary conditions, which we use to extract the faces across which we are to compute outflow. You can find details of how the periodic grid is constructed in the source code. With the modified grid in place, the loop for computing local flow solutions reads:

```
dp = Dp{1}*eye(d); nbcp = bcp;
for i=1:d
   for j=1:d,  nbcp.value(bcp.tags==j)=dp(j,i);  end
   xr = incompTPFA(initResSol(Gp, 100*barsa, 1), Gp, hT, fluid, 'bcp', nbcp);
   for j=1:d,
      v(j,i) = sum(xr.flux(ofaces{j})) / sum(Gp.faces.areas(ofaces{j}));
   end
end
```

Inside the loop over all axial directions, the first for-loop extracts the correct pressure drop to be included in the periodic boundary conditions from the diagonal matrix `dp`; that is, a pressure drop along the current axial direction and zero pressure drop in the other directions. Then, we solve the flow problem and compute the average velocity across the outflow boundaries. Because of the periodic conditions, we may have outflow also across boundaries that have no associated pressure drop, which is why we generally get a full-tensor permeability. Outside the loop, we compute the average pressure drop in each axial direction and invert Darcy's law to compute the permeability tensor:

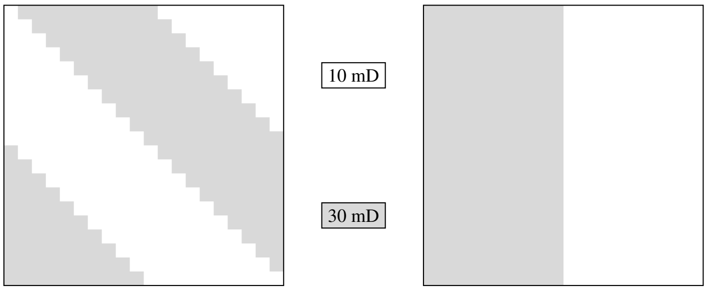
```
dp = bsxfun(@rdivide, dp, L);
K  = convertTo(v/dp, milli*darcy)
```

This implementation is offered as a utility `upscalePermeabilityPeriodic` in the `upscaling` module. Linear boundaries are not supported in MRST, but can be implemented by combining elements from the two cases just discussed: using the function `pside` to set a pressure distribution on the boundaries parallel to the direction of the applied pressure drop and solving a small linear system to compute the components of the effective permeability.

Setting up the necessary flow problems is relatively simple when the grid corresponds to just one coarse block. It is a bit more involved to do this upscaling efficiently for many blocks at a time. In the `upscaling` module, we therefore offer a utility function `upscalePerm` that computes permeability upscaling using the pressure-drop method.

Example 15.3.3 *Figure 15.7 shows two different permeability fields upscaled using flow-based upscaling with sealing and periodic boundary conditions. For the case to the right, the correlation in the permeability field is along the y-direction, and both methods compute the same diagonal upscaled tensor. Using pressure drop along the axial directions with sealing boundaries still gives a diagonal tensor if we rotate the grid so that the correlation direction is along the diagonal, whereas the periodic boundary conditions give a full tensor. By computing the eigenvalue decomposition of this tensor, we find that the upscaled tensor is diagonal with values 15 and 19.5 if we rotate the axial directions 45 degrees clockwise. Full source code for this example is given in the* `permeabilityExample1` *script.*

Example 15.3.4 *Both flow-based methods will by design correctly reproduce flow along or orthogonal to layered media for the experiment in Table 15.1. We therefore replace the layered permeability field with one having dipping layers. Moreover, we replace the top-to-bottom pressure drop by a pressure drop between diagonally opposite corners and sample from a different part of the SPE 10 model. Figure 15.8 shows the new permeability fields*



$$\mathbf{K}_s = \begin{bmatrix} 17.23 & 0 \\ 0 & 17.23 \end{bmatrix}, \mathbf{K}_p = \begin{bmatrix} 17.25 & -2.25 \\ -2.25 & 17.25 \end{bmatrix}, \qquad \mathbf{K}_s = \begin{bmatrix} 15.0 & 0 \\ 0 & 20.0 \end{bmatrix}, \mathbf{K}_p = \begin{bmatrix} 15.0 & 0 \\ 0 & 20.0 \end{bmatrix}$$

Figure 15.7 Upscaling two isotropic permeability fields using flow-based upscaling with sealing boundaries (s) or periodic boundary conditions (p).

Table 15.2 *Ratio between flow rate predicted by upscaled/fine-scale models for three different permeability fields, flow scenarios, and upscaling methods.*

Model	Flow pattern	Harm-arith	Sealing	Periodic
Layered	East→West	0.78407	1.00000	1.02594
	North→South	0.49974	1.00000	1.00000
	Corner→Corner	1.04273	1.30565	1.34228
Tarbert	East→West	0.86756	1.00000	0.56111
	North→South	0.89298	1.00000	0.53880
	Corner→Corner	0.00003	0.00027	39.11923
Upper Ness	East→West	0.83026	1.00000	0.40197
	North→South	0.71283	1.00000	0.28081
	Corner→Corner	0.09546	0.62377	2183.26643

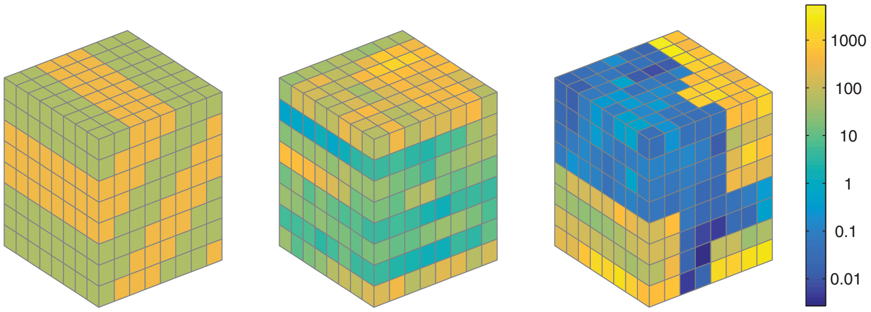


Figure 15.8 Three models used to test the accuracy of averaging techniques. From left to right: an isotropic model with dipping layers and two subsets from the Tarbert and the Upper Ness formation, respectively, from the SPE 10 data set.

and Table 15.2 reports the results of the experiment. The method with sealing boundary conditions is exact by design for the first two flow fields, but gives incorrect flow rates for the diagonal flow, in particular for the Tarbert subsample. With periodic boundary conditions, we get the exact flux when the flow direction follows regions of constant permeability from the north to the south side for the isotropic case. For the other two flow patterns, the flow directions cross dipping layers and we thus get minor deviations in the lateral direction and somewhat larger deviation for the diagonal flow. For the anisotropic subsamples, the periodic conditions generally give less accurate results than the other two methods, except for the diagonal flow on the Tarbert case. Altogether, this example illustrates quite well how ill-posed the upscaling problem is. Source code is given in the `permeabilityExample2` *script.*

Computer exercises

15.3.1 The data sets `BedModels1` and `BedModel2` are examples of fine-scale rock models developed for use in a workflow that propagates petrophysical properties from the core scale to the reservoir scale. Use the methods presented in this subsection to upscale the absolute permeability in these models.

15.3.2 Implement a method for upscaling with linear boundary conditions.

15.3.3 Estimates computed by the averaging and flow-based methods have a certain ordering, from low to high values: harmonic, harmonic-arithmetic, sealing boundaries, periodic boundaries, linear boundaries, arithmetic-harmonic, and arithmetic. Make a small test suite of heterogeneity permeability fields and verify this claim.

15.3.4 For grids that are not K-orthogonal, we know that the TPFA method will compute incorrect fine-scale flow solutions and that it generally is better to use a consistent discretization. Go back to Example 15.3.3, perturb the grid (e.g., with `twister`) and compare upscaled solutions computed by TPFA and the mimetic or MPFA method. (At the time of writing, the consistent methods do not yet implement periodic boundary conditions.)

15.4 Upscaling Transmissibility

In the discrete case, the choice of an appropriate upscaling method depends on the numerical stencil to be used for the spatial discretization of the upscaled model. In previous chapters we have seen that the two-point finite-volume method is the method of choice in reservoir simulation. When using this method, we only need grid-block permeabilities to compute transmissibilities between neighboring coarse blocks. It would therefore be more convenient if, instead of computing an effective permeability tensor, we could compute the coarse transmissibilities associated with the interface between pairs of neighboring coarse blocks directly. These transmissibilities should be defined so that they reproduce fine-scale flow fields in an averaged sense. That is, instead of upscaled block-homogenized tensors $\mathbf{K}^*$, we seek block transmissibilities T_{ij}^* satisfying

$$v_{ij} = T_{ij}^* \left(\frac{1}{|\Omega_i|} \int_{\Omega_i} p \, d\vec{x} - \frac{1}{|\Omega_j|} \int_{\Omega_j} p \, d\vec{x} \right), \tag{15.13}$$

where $v_{ij} = -\int_{\Gamma_{ij}} (\mathbf{K}\nabla p) \cdot \vec{n} \, d\nu$ is the total Darcy flux across Γ_{ij}.

We can compute the upscaled transmissibilities T_{ij}^* much the same way we computed upscaled permeabilities in Figure 15.9. Here, we use a pressure drop to drive a flow across the interface Γ_{ij} between two coarse blocks Ω_i and Ω_j. Thus, by solving (15.5) in the two-block domain $\Omega_i \cup \Omega_j$ subject to suitable boundary conditions, we can compute the average pressures P_i and P_j in Ω_i and Ω_j and then obtain T_{ij}^* directly from the formula

$$v_{ij} = T_{ij}^* (P_i - P_j). \tag{15.14}$$

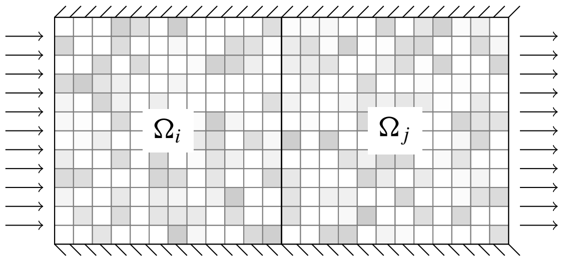


Figure 15.9 Illustration of flow-based upscaling of transmissibility.

Let us see how this upscaling can be implemented in MRST. That is, we discuss how to upscale the transmissibility associated with a single interface between two grid blocks. To this end, we assume a fine grid `G` and a coarse partition vector `q` that subdivides the grid into two coarse blocks in the x-direction. We start by constructing a coarse-grid structure `CG`, as introduced in Section 14.2. From the data members in this structure we can find all faces in the fine grid that lie on the interface between the coarse blocks and determine the correct sign to apply to the coarse-scale flux

```
i = find(~any(CG.faces.neighbors==0,2));
faces = CG.faces.fconn(CG.faces.connPos(i):CG.faces.connPos(i+1)-1);
sgn   = 2*(CG.faces.neighbors(i, 1) == 1) - 1;
```

As we saw in Section 15.3.2, there are different ways we can set up a localized flow problem that gives a flux across the interface between the two blocks. Assuming that the coarse interface is more or less orthogonal to the x-axis, we use a pressure drop in this direction and no-flow boundary conditions on the other sides, as shown in Figure 15.9:

```
bc   = pside([], G, 'XMin', Dp(1));
bc   = pside(bc, G, 'XMax', Dp(2));
xr   = incompTPFA(initResSol(G, 100*barsa, 1), G, hT, fluid, 'bc', bc);
flux = sgn * sum(xr.flux(faces));
mu   = fluid.properties();
```

All that now remains is to compute pressure values P_i and P_j associated with each coarse block before we use (15.14) to compute the effective transmissibility. Following (15.13), P_i and P_j are defined as the average pressure inside each block

```
P = accumarray(q,xr.pressure)./accumarray(q,1);
T = mu*flux/(P(1) - P(2));
```

Here, we have assumed that all grid cells have the same size. If not, we need to weight the pressure average by cell volumes. As an alternative, we could also have used the pressure value at the block centroids, which would give a (slightly) different transmissibility:

```
cells = findEnclosingCell(G,CG.cells.centroids);
P     = xr.pressure(cells);
```

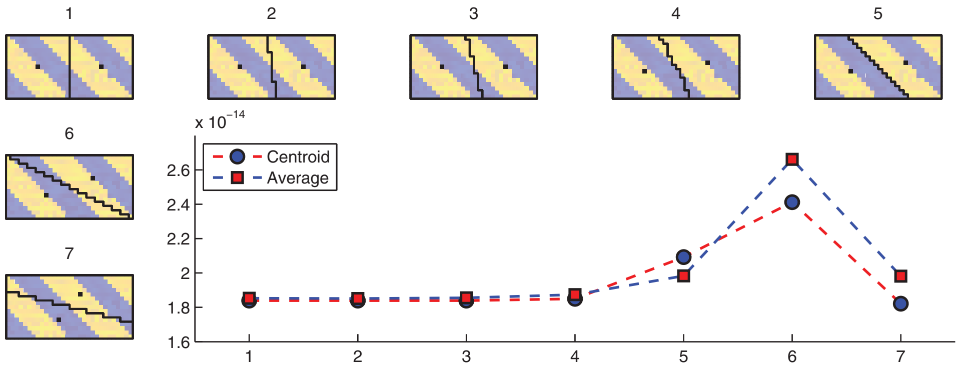


Figure 15.10 Upscaled transmissibility computed on a series of two-block configurations using either centroid values or averaged values for the block pressures. Black dots show the location of the block centroids from which pressure values are sampled.

Example 15.4.1 *In Figure 15.10 we have used the code just discussed to compute the transmissibility between two blocks covering a rectangular domain* $[0{,}200] \times [0{,}100]$ *m*^{2}*. We gradually rotate the interface between the two coarse blocks so that it goes from vertical towards being horizontal. Complete source code is given in the script* `transmissibilityExample1` *in the* `book` *module. For the first four pairs of coarse blocks shown in Figure 15.10, there is only a slight difference between the transmissibilities computed using pressure values defined as block averages and pressures defined by sampling at the block centroids. The transmissibilities start to deviate as the block interface approaches the diagonal.*

A pressure drop along the x-axis is a reasonable drive mechanism for the first block pairs, but it is highly questionable for the last two block pairs. Let us, for instance, consider block pair number seven. If we instead apply the pressure drop along the y-axis to create a flow that is more perpendicular to the coarse interface, the transmissibility computed with block-averaged pressures changes from 2.0e-14 to 8.1e-14, whereas the value is 8.9e-14 with a pressure drop along both axial directions. (The latter setup is generally not well-posed, as it will have singularities at the southeast and northwest corners.) Depending upon what type of flow conditions the block pair is subject to in subsequent simulations, it might be better to set a pressure drop orthogonal to the coarse-block interface or use one of the oversampling methods that will be introduced in the next section.

An important property of the upscaled transmissibilities T^*_{ij} is that they all should be positive across all interfaces that can transmit fluids, since this will ensure that the TPFA scheme defined by $\sum_j T^*_{ij}(p_i - p_j) = q_i$ reproduces the net grid block pressures $p_\ell = \frac{1}{|\Omega_\ell|}\int_{\Omega_\ell} p\,d\vec{x}$, and hence also the coarse fluxes v_{ij}. Unfortunately, there is no guarantee that the transmissibilities defined by (15.13) are positive, for instance, if the blocks are such that the chord between the cell centroids does not cross the coarse interface. Negative transmissibilities may also occur for regular block shapes if the permeability is sufficiently heterogeneous (e.g., as in the SPE 10 model). Even worse, unique transmissibility values may not even exist, since the upscaling problem generally is not well-posed; see [318] for a more thorough discussion of existence and uniqueness. To guarantee that the resulting coarse-scale discretization is stable, you should ensure that the T^*_{ij} values are positive. A typical trick of the trade is to set $T_{ij} = \max(T_{ij}, 0)$, or in other words, just ignore ill-formed connections between neighboring grid blocks. This is generally not a satisfactory solution, and you should therefore either try to change the grid, use different pressure points to define the transmissibility, or apply some kind of fallback strategy that changes the upscaling method locally.

The `upscaling` module does not contain any simple implementation of transmissibility upscaling like the one outlined in this section. Instead, the module offers a routine, `upscaleTrans`, that is designed for the general case where both the coarse and the fine grids can be fully unstructured and have faces that do not necessarily align with the axial direction. To provide robust upscaling for a wide range of geological models, the routine relies on a more comprehensive approach that will be outlined briefly in the next section, and also has fallback strategies to reduce the number of negative permeabilities.

15.5 Global and Local–Global Upscaling

The methods described so far in this chapter have all be local in nature. Averaging methods derive upscaled quantities solely from the local heterogeneous structures, whereas flow-based methods try to account for flow responses by solving local flow problems with prescribed boundary conditions. These boundary conditions are the main factor that limits the accuracy of flow-based methods. In Table 15.2, we saw how neither of the local methods were able to accurately capture the correct net flow for the anisotropic Tarbert and Upper Ness samples with boundary conditions giving diagonal flow. The main problem for flow-based techniques is, of course, that we do not know a priori the precise flow that will occur in a given region of the reservoir during a subsequent simulation. Thus, it is generally not possible to specify the appropriate boundary conditions for the local flow problems in a unique manner unless we already have solved the flow problem.

To improve the accuracy of the upscaling, you can use a so-called *oversampling technique*, which is sometimes referred to as an overlapping method. In oversampling methods, the domain of the local flow problem is enlarged with a border region that surrounds each (pair of) grid block(s) and boundary conditions, or other mechanisms for driving flow, like wells or source terms, are specified in the region outside the domain you wish to upscale. Local flow problems are then solved in the whole enlarged region, but the effective permeability tensor is only computed inside the original coarse block. This way, you lessen

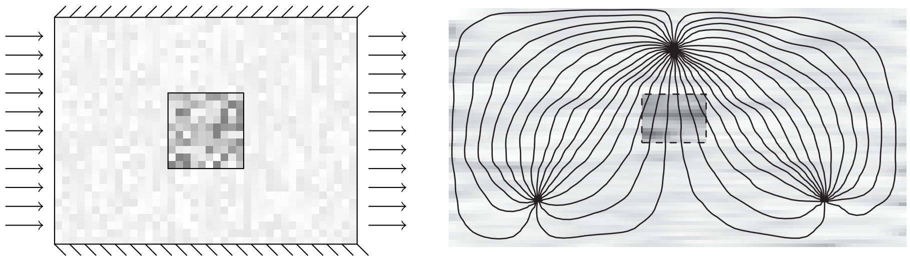


Figure 15.11 Illustration of oversampling techniques for upscaling. The size of the grid block is exaggerated in the right plot.

the impact of the localization assumption and the particular mechanism used to drive flow, and the flow across the boundaries of the coarse block is to a larger extent determined by the heterogeneity in the local region surrounding the block. The motivation for using an oversampling technique is to better account for permeability trends that are not aligned with the grid directions and possible large-scale connectivity in the permeability fields.

To also account for global flow patterns, you can solve the pressure equation once, or a couple of times, on the full geological model and impose fluxes sampled from the global flow solution as boundary conditions on the local flow problem used to upscale permeabilities or transmissibilities. In these so-called *global upscaling methods*, the pressure solution can either be computed using representative and *specific* drive mechanisms. Alternatively, you can compute the pressure solution using a set of *generic conditions* that, e.g., specify a flow through the reservoir from east to west, from north to south, and so on. From a computational point of view, this approach may seem to contradict the purpose of upscaling, but can be justified for numerical simulation of compressible and/or multiphase flow. Indeed, for such transient flows, the pressure equation must be solved multiple times throughout the simulation, and the cost of solving the global pressure equation once (or even a few times) on a fine grid will in most cases be small compared with the cost of solving the full multiphase flow problem.

Computing a global pressure solution on the fine grid can be avoided if we use a so-called *local–global* method [64, 63, 117]. In these methods, pressure solutions obtained by solving global flow problems on a coarse grid are used to set boundary conditions for local flow problems that determine the upscaled transmissibilities. Since the initial coarse-grid computation may give quite poor boundary conditions for the local flow problems, these methods use a iterative bootstrapping procedure to gradually ensure consistency between the local and the global calculations.

Global and local–global upscaling methods may suffer from negative or anomalously large transmissibility values. To avoid these, Holden and Nielsen [133], who were among the first to study global upscaling methods, proposed an iterative process involving the solution of an optimization problem that gradually perturbs the transmissibilities until they fall within a prescribed interval. A similar technique that avoids the solution of the fine-grid problem was presented in [225]. Chen and Durlofsky [63] observed that unphysical transmissibilities occur mainly in low-flow regions. Therefore, instead of using an optimization procedure that alters all transmissibilities in the reservoir, they proposed to use a thresholding procedure in which negative and very large transmissibilities are replaced by transmissibilities computed by a local method. Because the transmissibilities are altered only in low-flow regions, the perturbation will have limited impact on the total flow through the reservoir.

The script `upscaleTrans` offers various implementations of global transmissibility upscaling using either specific or generic mechanisms to drive flow in the full reservoir model. This specific approach enables you to tailor the coarse-scale model and achieve very good upscaling accuracy for a particular simulation setup. The generic approach gives you a more robust upscaling that is less accurate for a specific simulation setup, but need not be recomputed if you later wish to simulate setups with significant changes in the well pattern, aquifer support, and other factors that affect the global flow patterns in the reservoir. Local–global methods are not yet supported in MRST. Instead, the software offers various multiscale methods [95], including multiscale mixed finite elements [6, 243], multiscale finite volumes [211, 214], and the multiscale restriction-smoothed basis method [216, 215]. These methods offer an alternative approach, in which the impact of fine-scale heterogeneity is included more directly into coarse-scale flow equations as a set of basis functions that not only represent the average effect but also incorporate small-scale variation. The methods also give a natural way to reconstruct fine-scale pressure and mass-conservative flux approximations.

Computer exercises

15.5.1 Extend the computations reported in Table 15.2 to include oversampling methods. Does this improve the accuracy of the two flow-based methods?

15.5.2 Implement a function that performs generic global upscaling for uniform partitions of rectilinear and curvilinear grids. To drive the global flow pattern, you can use a pressure drop in each of the axial directions. How would you extend your method to more general grids and partitions?

15.5.3 Extend the `upscaling` module to include local–global upscaling methods.

15.6 Upscaling Examples

In this section we go through three examples that apply the methods discussed earlier in the chapter to upscale reservoir models. In doing so, we also introduce you to a simple form of flow diagnostics [274, 218] you can use to assess the accuracy of single-phase upscaling. The last example represents a complete workflow, from geological surfaces to grid, via fine-scale simulation to upscaled model. We end the section with a set of general advice and simple guidelines.

15.6.1 Flow Diagnostics Quality Measure

A challenge with upscaling is to predict upfront whether the upscaling will be accurate or not. To partially answer this question and give an indication of the quality of the upscaling, we suggest to compare the cumulative well-allocation factors computed by the fine-scale and upscaled models. As shown in Section 13.1.3, a well-allocation factor is the percentage of the flux in or out of a completion that can be attributed to a pair of injection and production wells. This information is similar to what is obtained by a production logging tool.

To compute these factors, we need to first compute a global, single-phase pressure solution for the specific well pattern we want to investigate. From the resulting flow field we compute influence regions that subdivide the reservoir into subvolumes associated with pairs of injection and production wells, as discussed in Section 13.1. Global methods tend to compute at least one fine-scale pressure solution as part of the upscaling procedure, but if a global solution is not available for the specific well pattern, it is generally not very expensive to compute compared with the cost of the flow-based upscaling procedure. The same goes for the computation of a coarse-scale flow field and partition functions on the fine and the upscaled model.

To improve the predictive power of well-allocation factors, you may have to subdivide each well into two or more segments that each consists of a connected set of completed cells. This way, you can measure the communication between different parts of the wells, which is particularly important if the model to be upscaled includes layers or geological objects with significantly different permeabilities. You can obviously also investigate to what extent dynamic heterogeneity measures, sweep/drainage regions, and time-of-flights are preserved, or compare the fine and coarse fluxes to assess the accuracy of the upscaling.

15.6.2 A Model with Two Facies

We start by considering a small rectangular reservoir containing two facies with contrasting petrophysical properties. The reservoir is produced by an injector and a producer placed diagonally opposite each other and completed mainly in the high-permeability parts of the reservoir. Both wells operate at a fixed rate that amounts to the injection/production of 0.2 pore volumes per year. We neglect gravity forces to avoid potential complications from cross-flow. The fine grid consists of $40 \times 20 \times 15$ cells, which we seek to upscale to a coarse $5 \times 5 \times 15$ model. That is, we upscale by a factor 8×4 in the lateral direction and leave the vertical layers in an attempt to preserve the vertical communication in the model as accurately as possible. Figure 15.12 shows the porosities of the fine and the coarse model. To also upscale permeability, we use `upscalePerm`, which implements the flow-based method from Section 15.3.2 with sealing boundary conditions. The script `upscalingExample1` in the `book` module contains complete source code.

To get an indication of the accuracy of this upscaling, we compare the volumetric connections between the injector and producer computed for the original and the upscaled permeability. (The latter is prolongated back onto the fine grid.) With a single

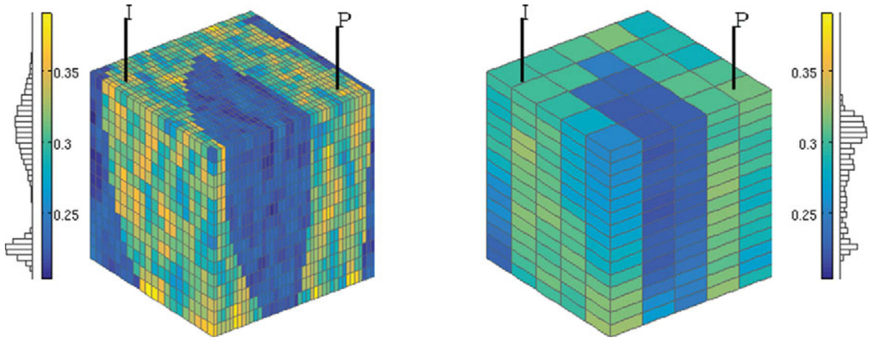


Figure 15.12 Porosity distribution in a model with two different rock types before and after upscaling.

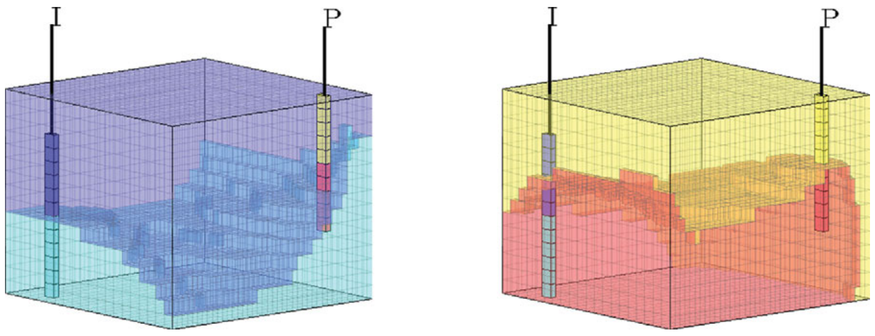


Figure 15.13 Volumetric subdivision of flooded/drainage regions by majority vote. The left plot shows the regions flooded by the upper and lower parts of the injector in blue and cyan, respectively. The right plot shows the drainage regions of the upper and lower half or the producer in yellow and red, respectively. In both plots, each cell is assigned to the completion that has the highest concentration value.

injector–producer pair that operates at fixed rates, the only difference we can expect to see is variation in the distribution of flow rates within each well. To better measure the communication within the reservoir, we therefore subdivide each well into an upper and a lower segments, so that we altogether have four well-pairs, whose well-allocation factors can be used to verify the quality of the upscaling. Figure 15.13 shows the influence regions of the fine-scale model for each of the four segments, defined as regions with "color concentration" larger than 0.5. A good upscaling method should preserve these communication volumes as accurately as possible in the coarse model.

Volumes are not always easy to compare in complex 3D models. Instead, we compare cumulative well-allocation factors. The plots in Figure 15.14 show bar charts of the cumulative flux in/out of the completions that make up each segment, from bottom to top. Here, each segment is assigned the same unique color as in Figure 15.13 and each bar is subdivided into the fraction of the total influx/outflux that belongs to the different well-pairs the segment is part of; see Section 13.1.3. To compare the fine-scale and coarse-scale permeability, **K** and **K***, we use colors for well allocation computed with the *coarse* model, whereas solid lines show the same quantities computed for the fine-scale model. The closer the color bars and the solid lines are, the better the upscaling.

To explain how this type of flow diagnostics should be interpreted, let us look at the lower-left bar chart, which reports the allocation factors for the upper segment of the producer (`P:1`, yellow color). Here, blue color signifies inflow that can be attributed to injection from the upper segment `I:1` of the injector, whereas cyan color signifies inflow attributed to the lower segment `I:2`. Because the colored bars are mostly blue, the inflow into the upper producer segment is predominantly associated with the `I:1`–`P:1` pair. In other words, `P:1` is mainly supported by the upper segment of the injector and is hardly connected to the lower segment. You can confirm this by looking at the upper-right bar chart, which shows that the flow from the lower injector segment mainly contributes to inflow in the lower segment of the producer. The flooded volumes shown to the left in Figure 15.13 show that fluid injected from `I:2` will mainly sweep the volume surrounding the lower three cells of `P:2`. Similarly, the right plot shows that the drainage region of `P:1` only engulfs the upper parts of `I:1`. The flooded regions of `I:1`, on the other hand, engulfs both `P:1` and the upper parts of `P:2`, and hence the upper injector segment will contribute flux to both segments of the producer, as shown in the upper-left chart of Figure 15.14. Likewise, the drainage region from `P:2` engulfs both `I:2` and the lower part of `I:1` and hence is supported by flux from both the injector segments, as shown in the lower-right chart of Figure 15.14.

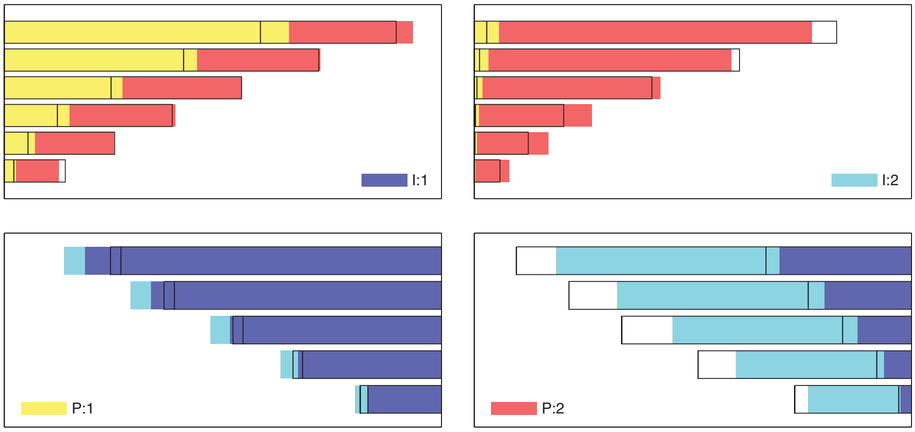


Figure 15.14 Normalized cumulative well-allocation factors for the four well completions in the two-facies model. The color bars show well-allocation factors computed with **K*** (upscaled by `upscalePerm`) on the fine grid, while the black lines show the same factors computed with the original permeability **K**. The colored bars and the black lines should coincide if the upscaled permeability reproduces the flux allocation correctly.

So, what does this flow diagnostics tell us about our upscaled permeability? First of all, $\mathbf{K}^*$ does not fully reproduce the volumetric connections of the original permeability $\mathbf{K}$. Looking at well-allocation factors for the producers in the bottom row of Figure 15.14, we see that $\mathbf{K}^*$ predicts too much flux in the upper segment (`P:1`, yellow color, left chart) since the colorbars exceed the solid lines at the top. Likewise, in the lower segment of the producer (`P:2`, red color, right chart), we get too low influx. Altogether, the allocation factors seem to suggest that the main problems are that with $\mathbf{K}^*$ we miss some of the vertical communication and exaggerate the connection between `I:1` and `P:1`. We could have used a single number to measure the overall discrepancy in flux allocation, but in my opinion you will benefit more from a graphical presentation like this for problems with only a few wells, because it may help you identify the cause of the mismatch between the upscaled and the original permeability. (Lorenz coefficients for $\mathbf{K}$ and $\mathbf{K}^*$ are 0.2329 and 0.2039, respectively.)

To develop a better upscaling strategy, we can upscale transmissibilities and well indices using a global method. For a given flow scenario, like the one with fixed well positions and constant well controls, you can determine a set of transmissibilities and well indices that reproduce a single pressure step exactly. This may seem desirable, but has the disadvantage that the linear system becomes negative definite, which in turn may lead to unphysical solutions away from the scenario used for upscaling. To span a reasonable set of global flow directions, we instead use generic boundary conditions that pressurize the reservoir from east to west, from north to south, and from top to bottom. We can then combine these three global flow fields in several different ways to compute a transmissibility value for each coarse face; see e.g., [152]. Here, we use a simple strategy suggested by [172]: We pick the one among the three flow field that gives the largest flux across a given coarse face and insert this into (15.14) to compute the transmissibility associated with the face. A rationale for this choice is that the accuracy of the coarse model is most sensitive to the choice of transmissibilities in high-flow regions.

We upscale *well indices* using a coarse-scale version of the inflow-performance relationship (see (4.27) on page 122)

$$J_i^* = Q_i/(P_i - p_w), \tag{15.15}$$

where Q_i is the sum of the rates of all the perforations inside block i, P_i is the average block pressure, and p_w is the pressure inside the wellbore. To obtain representative flow fields for n wells, `upscaleTrans` solves $n-1$ pressure equations. Each solution is computed by setting a positive pressure in one well and zero pressure in all the other. We can then combine the flow fields using the same strategy as for the transmissibilities by picking the solution that gives the largest well rate, or as we will do here, add all well rates and divide by the sum of the corresponding pressure drops. (For models with many wells, we would rather upscale well indices using local or specific approaches.)

Unfortunately, there is no guarantee that the upscaled transmissibilities are nonnegative. To avoid creating an ill-conditioned discretization matrix, we therefore set all negative transmissibilities to zero, thereby blocking the corresponding coarse face for flow. This

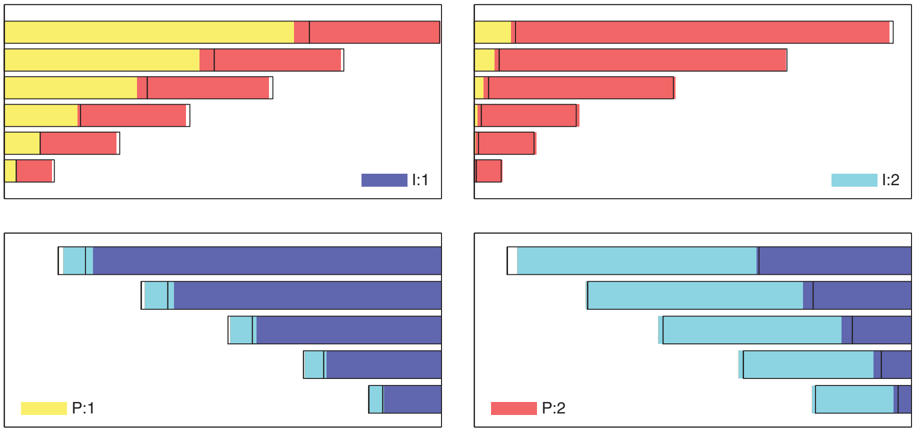


Figure 15.15 Normalized cumulative well-allocation factors for the four well completions in the two-facies model. The color bars show well-allocation factors computed on a coarse model upscaled using `upscaleTrans`, while the black lines show the same factors computed on the original model.

is in our experience an acceptable solution as long as the upscaled transmissibilities are only intended for a specific scenario, like herein, since negative transmissibilities tend to appear because flow is tangential to the coarse interface or as a result of numerical noise in regions with very low flow. If your upscaling, on the other hand, aims to serve a more general purpose with varying well patterns and/or boundary conditions, you would need to implement a more sophisticated *fallback strategy* that recomputes negative transmissibilities by an alternative method like a local flow-based method, an averaging method, or a combination of these.

Figure 15.15 shows the resulting match in well-allocation factors, which is significantly better than when using a simple permeability upscaling. In part, this is a result of improved transmissibilities, and in part because we now have upscaled the well indices.

15.6.3 SPE 10 with Six Wells

You have already encountered Model 2 from the 10th SPE Comparative Solution Project multiple times throughout the book. Because of its simple grid and strong heterogeneity and the fact that it is freely available online, this data set has become a community benchmark that is used for many different purposes. The original aim of the project was to "*compare upgridding and upscaling approaches and the ability to predict performance of a waterflood through a million cell geological model*," and in [71] you can read about the relative merits of the various methods used in the upscaling studies that were submitted to the project by August 2000. In short, the study showed that the data set is generally very difficult to upscale accurately with single-phase methods and that the best results are

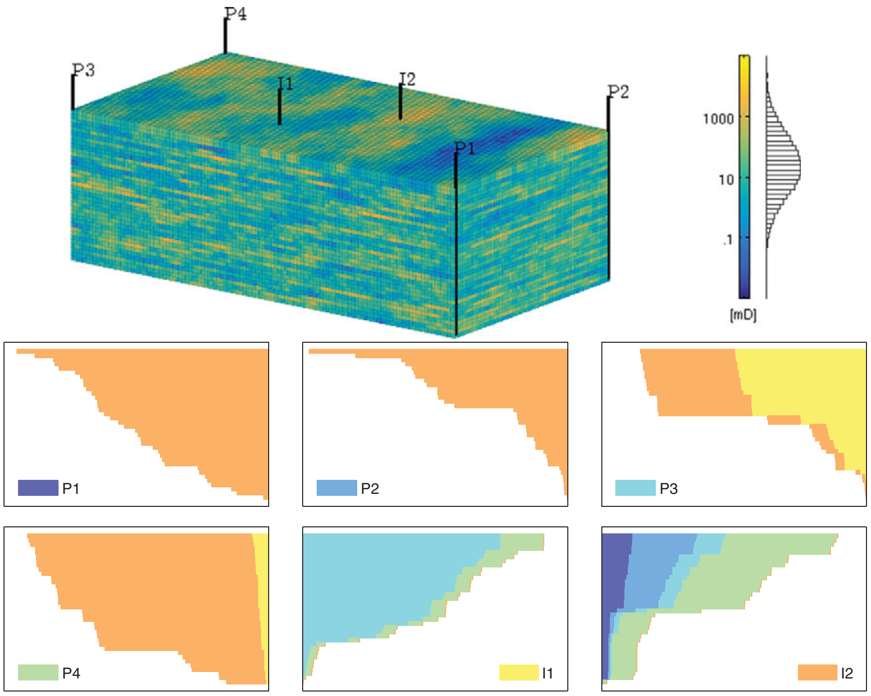


Figure 15.16 Model of the Tarbert formation with six wells, two central injectors and four peripheral producers. The 3D plot shows lateral permeability distribution, whereas the bar charts show normalized cumulative well-allocation factors.

generally obtained by methods that also account for multiphase effects. Later, however, several authors have obtained excellent results with local–global and multiscale methods.

In this section, we compare how accurate four different upscaling methods can predict single-phase flow. That is, we consider two averaging methods (harmonic and harmonic-arithmetic) and a local flow-based upscaling with sealing boundary conditions for upscaling permeability, and the global upscaling method discussed at the end of the previous example for upscaling transmissibilities and well indices. Unlike in the previous example, we use maximum flow rate to determine both transmissibilities and well indices. The script `upscalingExample2` contains full details.

We start by considering the Tarbert formation, which is found in the upper 35 layers of the model. We choose a coarsening factor of $10 \times 10 \times 3$, and to get an even number of cell layers inside each grid block, we extend the model slightly by repeating the top layer so that the fine-scale model altogether has $220 \times 60 \times 36 = 475\,200$ cells. We also replace the original five-spot pattern with a pattern consisting of producers in the four corners of the model and two injectors located near the center of the model; see Figure 15.16. All wells are controlled by bottom-hole pressure; the producers operate at 200 bar and the producers at 500 bar.

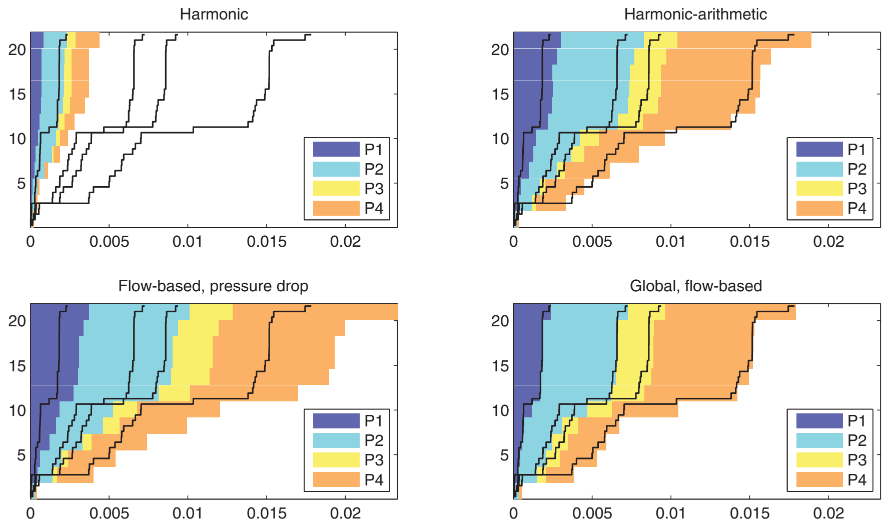


Figure 15.17 Well-allocation factors for injector I2 in the Tarbert model computed on four different coarse models.

Figure 15.17 compares well-allocation factors predict by the four upscaling methods for injector I2. This well has significant communication with all four producers and is hence more difficult to match than injector I1, which mainly communicates with P3 and P4. Harmonic averaging not surprisingly gives well rates that are far from the fine-scale targets. As explained in Section 15.3.1, the harmonic mean has a bias against smaller values and tends to underestimate the effective permeability. Because all wells are controlled by bottom-hole pressure, too low permeability means too low well rates. If the wells were controlled by total rate, as in the previous example, the well rates predicted by the harmonic average would have been more correct, but the predicted pressure buildup around the injector and the pressure drawdown in the producers would both be too high.

Comparing the local flow-based method to the harmonic-arithmetic averaging method, we see that the latter is significantly more accurate. By design, the flow-based method should give accurate prediction of flow along the axial directions of the grid. Here, however, we have flow that is strongly affected by heterogeneity and mostly goes in the lateral, diagonal direction and hence the harmonic-arithmetic method seems to be more suited, in particular if we consider the computational time of the two methods. In the current implementation, the flow-based method extracts a subgrid and assembles and inverts three local matrices for each coarse block. The computational overhead of this procedure is significant since each new local solve incurs full start-up cost. The result is that the function `upscalePerm` has orders of magnitude higher computational cost than the harmonic-arithmetic averaging, which is implemented using a few highly efficient calls to `accumarray`.

The global upscaling method reproduces the fine-scale well-allocation factors almost exactly. The plots may be a bit deceiving because of the areas where the colorbars extend beyond the solid lines representing the fine-scale well-allocation. However, bear in mind that the bars and lines represent *cumulative* factors, and thus we should only look at the discrepancy at the top of each colorbar. If you look closely, you will see that here the match is excellent. There are two reasons for this: First of all, with global boundary conditions, the boundary conditions used to localize the computation of each transmissibility account better for correlations that extend beyond the block pair and thus set up a flow that is more representative of the flow the interface will experience in the subsequent flow simulation. Secondly, the global method includes upscaling of well indices. This can have a significant impact on the well rates that determine the flux allocation. In passing, we also note that the computational cost of the global method is significantly lower than that of the local method, primarily since our implementation utilizes a highly efficient multigrid solver for the global flow problems and avoids solving a long sequence of small problems that each have a large start-up cost.

For completeness, we also include the results of a similar study for the whole SPE 10 model; see Figure 15.18. This includes the fluvial Upper Ness formation, which is notoriously difficult to upscale accurately. The harmonic-arithmetic and the global method are no longer as accurate as for the Tarbert formation. For this 1.1 million grid cell model, an efficient iterative solver like AGMG is indispensable to be able to compute the fine-scale solution and perform a global upscaling in MATLAB.

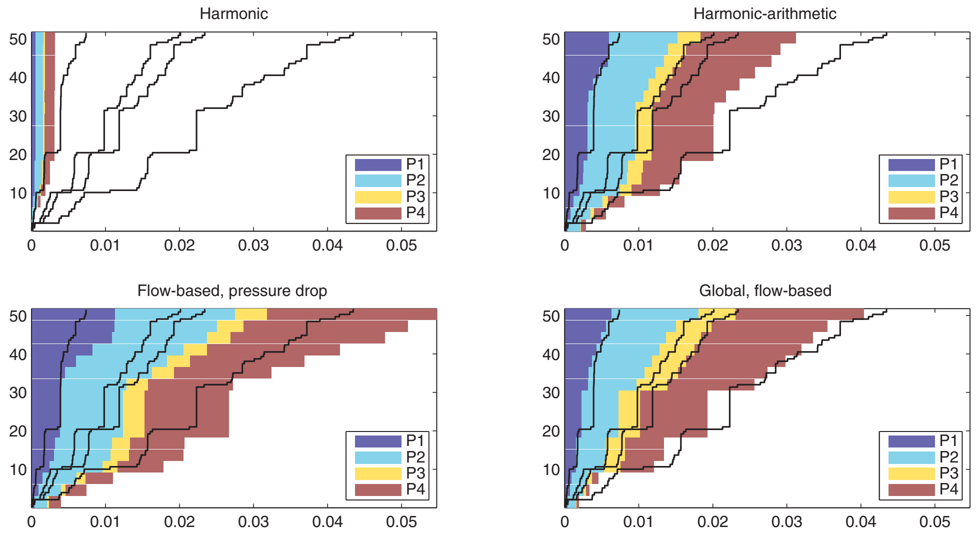


Figure 15.18 Well-allocation factors for injector I2 in the full SPE 10 model computed on four different coarse models.

15.6.4 Complete Workflow Example

Our last example shows a complete workflow for a synthetic sector model, from setting up the geocellular model, via compressible flow simulations, to coarsening/upscaling and analysis of the results using flow diagnostics. The example can thus be seen as a partial recap of the content in this book. We therefore include more code details than in the two previous examples. You can find the complete source code in `upscalingExample3` in the `book` module.

The reservoir has a layered stratigraphy and contains three faults. The reservoir fluids form a slightly compressible two-phase oil/water system and hydrocarbons are recovered by a well pattern consisting of one injector and three producers. The model is mostly conceptual and is designed to outline and give details of the workflow rather than as a problem representing a realistic scenario in terms of well locations and fluid physics. However, unlike many other examples in the book, it is completely self-contained and does not involve any ECLIPSE files to describe reservoir geometry, wells, fluid properties, and simulation schedule.

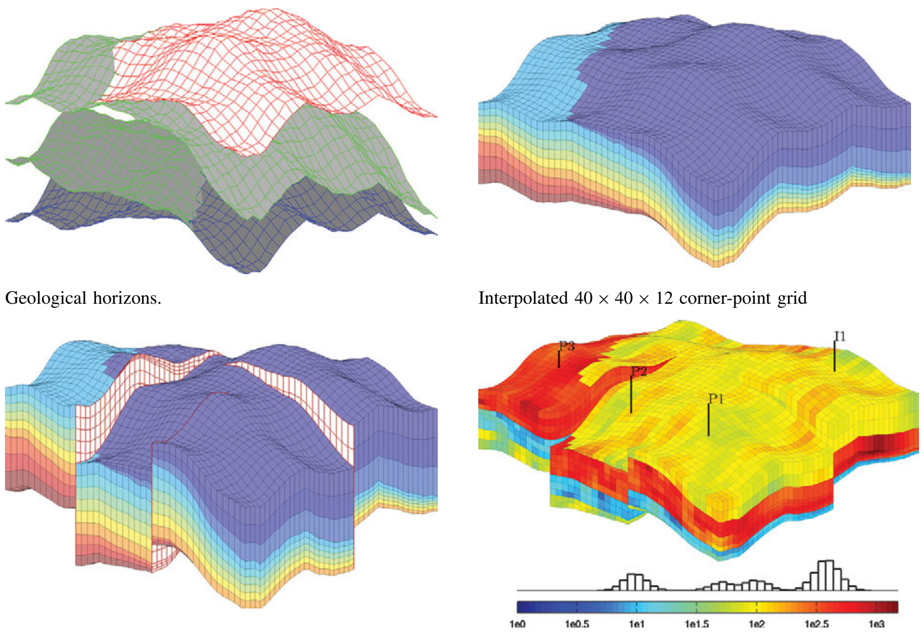
We start by creating the main horizons that deliminate the external and internal geology of the sector. The top surface is composed of three different trends: a large-scale anticline structure in the form of an elongated arc-like shape, a set of medium-scale sinusoidal folds, and a set of small-scale random perturbations. For simplicity, we use the same surface, shifted by a constant factor, to deliminate the reservoir downward. In between, we introduce two tilted surfaces with extra small-scale random perturbations. These surfaces are clipped against the top and bottom surface to create erosional effects. The four horizons are shown in Figure 15.19. Assuming that these surfaces are given in the arrays `zt`, `zmt`, `zbt`, and `zb`, we create a basic corner-point grid as follows (see Section 3.3.1):

```
horizons = {struct('x', x, 'y', y, 'z', zt), ...
  struct('x', x, 'y', y, 'z', zmt), ...
  struct('x', x, 'y', y, 'z', zmb), ...
  struct('x', x, 'y', y, 'z', zb)};
grdecl = convertHorizonsToGrid(horizons, 'dims', [40 40], 'layers', [3 6 3]);
```

To introduce the faults, we first extract the coordinates for the corner-points and then shift the z-coordinates in selected parts of the domain:

```
[X,Y,Z]  = buildCornerPtNodes(grdecl);
i=47:80; Z(i,:,:) = Z(i,:,:) + .022*min(0,Y(i,:,:)-550);
j= 1:30; Z(:,j,:) = Z(:,j,:) + .021*min(0,X(:,j,:)-400);
j=57:80; Z(:,j,:) = Z(:,j,:) + .023*min(0,X(:,j,:)-750);
grdecl.ZCORN = Z(:);
```

The lower-left plot in Figure 15.19 shows the resulting model. The permeability consists of four layers of lognormally distributed values with average lateral permeability of 100, 400, 10, and 50 md, from top to bottom. The vertical permeability is set to one-tenth of the lateral permeability.



Geological horizons. Interpolated $40 \times 40 \times 12$ corner-point grid

Introducing four partial faults. Lateral permeability and well pattern.

Figure 15.19 Creating a synthetic sector model whose geology is described by three volumes (two of which are eroded), three vertical faults that partially penetrate the sector, and four layers of distinctively different permeabilities. Oil is recovered by three producers supported by one injector.

The three vertical producers are completed in all layers of the model and operate at a constant bottom-hole pressure of 250 bar. The injector is also perforated in all layers and operates at a constant rate that would fill one pore volume of water over a period of ten years. As you probably recall, such wells are set up as follows:

```
W = verticalWell([], G, rock, 10, 10, [],...
            'Name', 'P1', 'comp_i', [1 0], 'Val', 250*barsa, ..
            'Type', 'bhp', 'refDepth', 50);
```

The fluid system is assumed to consist of two phases, an incompressible water phase and an oil phase with constant compressibility. If we express the oil compressibility as an expansion factor on the form, $b_o(p) = b_0 \exp[c(p - p_0)]$, we can model the fluid system as a special case of the general three-phase black-oil model from the AD-OO framework discussed in Chapter 12. The simplest fluid object in this framework models an incompressible fluid with constant densities. To get the correct behavior for the oil phase, we simply replace the function handle in the `bo` field by a pointer to an anonymous function that evaluates the `exp` function with appropriate arguments. We then call the appropriate constructor from the AD-OO framework:

```
fluid = initSimpleADIFluid('mu',    [1, 5, 0]*centi*poise, ...
                           'rho',   [1000, 700, 0]*kilogram/meter^3, ...
                           'n',     [2, 2, 0]);
fluid.bO = @(p, varargin) exp((p/barsa - 300)*0.001);

gravity reset on
model = TwoPhaseOilWaterModel(G, rock, fluid);
```

Initially, the reservoir is assumed to be in an equilibrium state with a horizontal oil–water (OW) contact separating pure oil from pure water. As explained earlier, MRST uses water, oil, and gas ordering internally, so in this case we have water in the first column and oil in the second for the saturations in the initial state

```
region = getInitializationRegionsBlackOil(model, depthOW, ...
           'datum_depth', ddepth, 'datum_pressure', p_ref);
state0 = initStateBlackOilAD(model, region);
```

The initialization is quite simple and does not perform any sub-cell integration to assign more accurate saturations in cells intersected by the OW contact; see the upper-left plot in Figure 15.20.

We define a straightforward simulation schedule consisting of five small initial control steps followed by 25 larger steps. We keep the well controls fixed throughout the simulation. If `dT` contains the time step, the schedule is constructed by the call `simpleSchedule(dT, 'W', W)`. The simulation model has 16,350 grid cells and the resulting linear systems are a bit too large for MATLAB's direct solver to be efficient. We therefore set somewhat stricter tolerances and use a CPR preconditioner with an algebraic multigrid solver for the elliptic pressure system as discussed in more detail for the SPE 9 benchmark in Section 12.4.4. Figure 15.20 shows how water displaces oil over time inside the reservoir.

To reduce computational times, we partition the original $40 \times 40 \times 12$ grid uniformly in index space into a new $20 \times 20 \times 4$ coarse grid. This will generally give blocks having cells on opposite sides of faults. To ensure that as many as possible of the coarse blocks are hexahedral (except for those that are partially eroded along the top or bottom), we split any coarse block intersected by one of the faults. To do this, we introduce a temporary grid topology in which faults are set as barriers, and then perform a simple postprocessing to split any disconnected blocks:

```
cdims = [20, 20, 4];
Gf = makeInternalBoundary(G, find(G.faces.tag > 0));
p = processPartition(Gf, partitionUI(G, cdims));
```

Altogether, this produces a coarse grid with 1,437 active blocks (see Figure 15.21), which corresponds to a little more than a 22 times reduction in the number of unknowns. This should put us well within the range of models for which we safely can use MATLAB's default linear solver without the need for a CPR preconditioner.

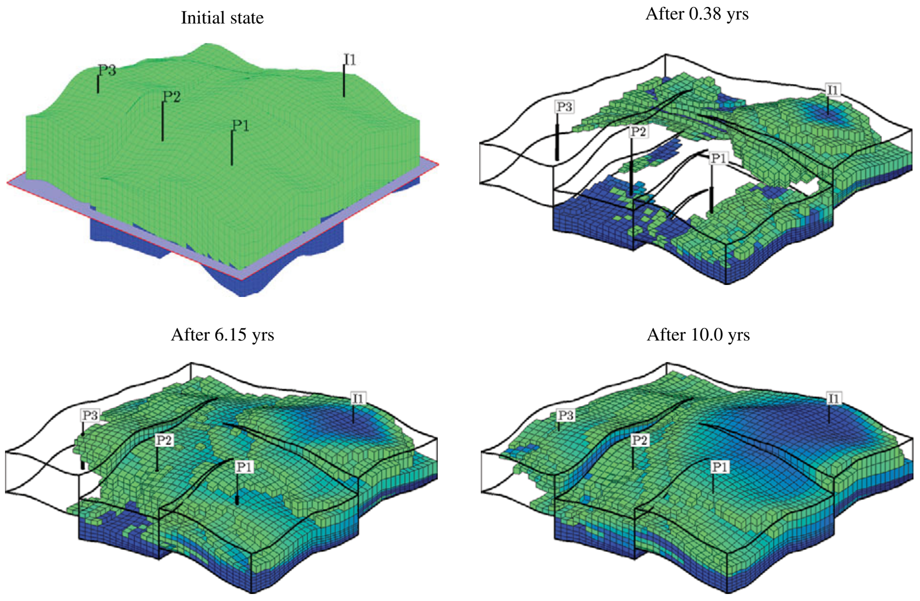


Figure 15.20 Time evolution of the water saturation for the sector model.

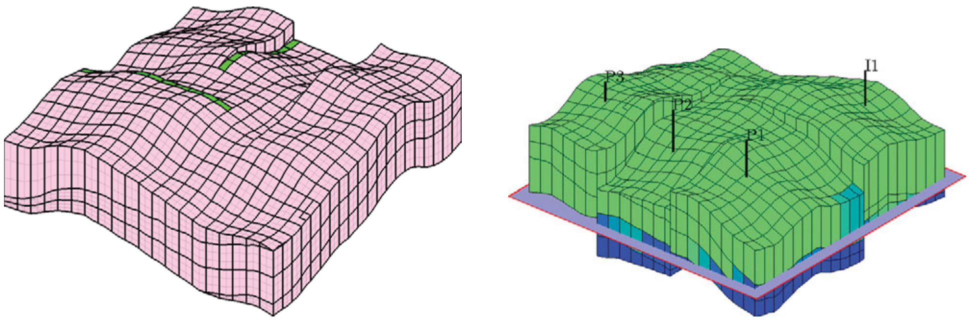


Figure 15.21 A coarse 20 × 20 × 4 partition of the sector model. New blocks arising when we split blocks penetrated by one of the faults are marked in green. The right plot shows the initial saturation as computed by `upscaleState`.

We can now directly upscale the model, schedule, and initial state. By default, the upscaling routine uses the simplest possible options, i.e., harmonic averaging of permeabilities,

```
modelC1    = upscaleModelTPFA(model, p);
scheduleC1 = upscaleSchedule(modelC1, schedule);
state0C1   = upscaleState(modelC1, model, state0);
```

We have already seen that power-averaging of permeability is not very accurate, and thus we only consider this model as a base case. In addition, we use flow-based transmissibility upscaling with a specific flow field obtained by solving a single-phase flow problem with the given well pattern. This approach has proved to be the most accurate in the examples earlier in the section:

```
[~, TC, WC] = upscaleTrans(GC, model.operators.T_all, ...
  'Wells', W, 'bc_method', 'wells', 'fix_trans', true);

modelC2 = upscaleModelTPFA(model, p, 'transCoarse', TC);
scheduleC2 = schedule;
for i = 1:numel(scheduleC2.control)
   scheduleC2.control(i).W = WC;
end
```

To get an idea of how accurate the two upscaling methods will be, we can compare well allocation with the fine model based on the computation of a single pressure step. The plots in Figure 15.22 confirm that harmonic averaging also in this case underestimates the coarse-scale permeability and hence will predict too small flux into the three production wells. Specific transmissibility upscaling, on the other hand, reproduces the correct flux allocation very accurately and it therefore seems likely that it will produce quite accurate results. Let us now simulate the two cases. This is done as for a standard fine-scale model by calling the standard simulator interface with the coarsened model, initial data, and schedule

```
[wellSolsC1, statesC1] = simulateScheduleAD(state0C1, modelC1, scheduleC1);
```

Somewhat surprising, the production curves predicted by the specific flow-based upscaling are generally not much more accurate than those predicted by the simple harmonic

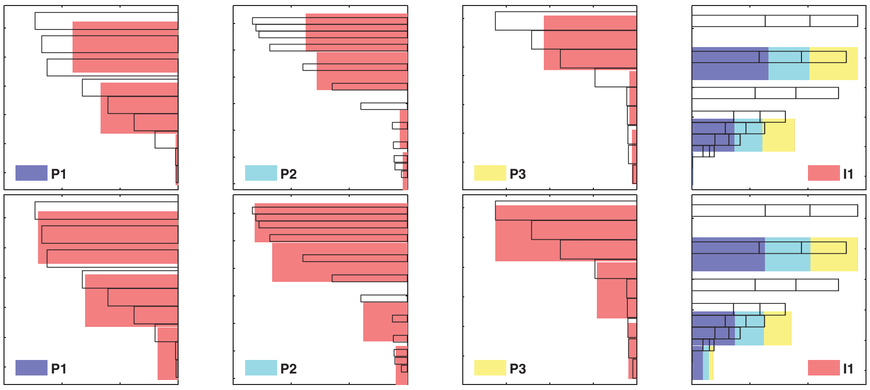


Figure 15.22 Flow diagnostics used as a quality measure for the two upscaling methods on the $20 \times 20 \times 4$ upscaled sector model with harmonic upscaling of permeability (top) and specific flow-based transmissibility upscaling (bottom).

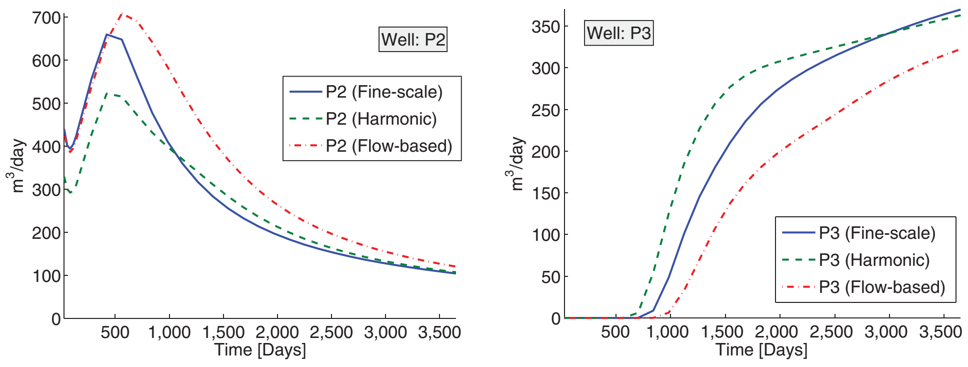


Figure 15.23 Comparison of oil rate (producer P2) and water rate (producer P3) predicted by the upscaled 20 × 20 × 4 models and the original 40 × 40 × 12 sector model.

permeability upscaling. Figure 15.23 shows the two production curves for which specific transmissibility upscaling compares least favorably with harmonic upscaling. This illustrates an important point: *whereas accurate reproduction of single-phase well allocation often is a* necessary *condition for accurate upscaling, it is not a* sufficient *condition*.

The question now is what has gone wrong here? The primary problem is that the grid blocks in the coarse grid do not adapt to the layering structure in the permeability field if we use a uniform subdivision of the K index. You may recall we argued that adapting to permeability layering is important when we discussed CaseB4 in Section 14.3.3. So let us rerun the upscaling exercise with a new basic partition that respects the layering

```
p0 = partitionLayers(G, cdims(1:2), L);
```

Here, `L=[1,3,8,11,13]` is a run-length encoding of the layer indices; this vector can be obtained as the second output of the `lognormLayers` function we used to generate the permeability field. The resulting coarse partition has exactly the same number of grid blocks, but gives significantly more accurate predictions for both upscaling methods, as you can see in Figure 15.24. We also obtain almost the same accuracy if we instead use a uniform 10 × 10 × 12 partition, which supports our hypothesis that preserving layering is very important in this particular example.

Computer exercises

15.6.1 How coarse model are you able to make and still predict the oil and water rates with reasonable accuracy?

15.6.2 Make a fine-grid model that respects the OW contact exactly. How does this affect the simulation? How would you coarsen and upscale this model?

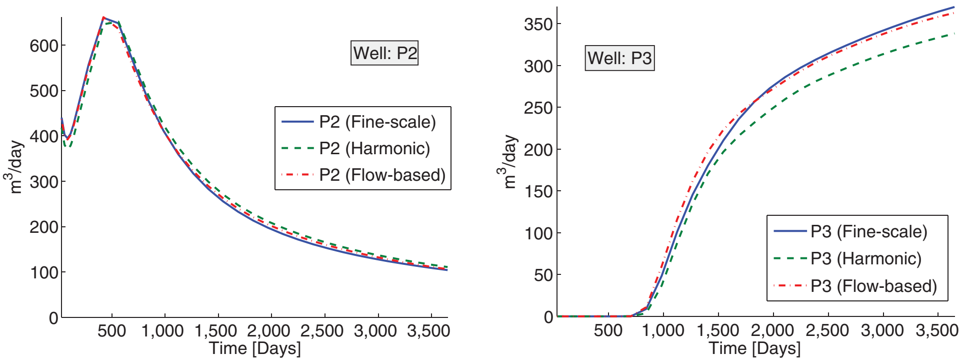


Figure 15.24 Comparison of oil rate (producer P2) and water rate (producer P3) predicted by the upscaled 20 × 20 × 4 models where the coarse blocks adapt to the permeability layers of the original 40 × 40 × 12 sector model.

15.6.5 General Advice and Simple Guidelines

Through the various examples in this chapter, I have tried (and hopefully succeeded) to convince you that accurate and robust upscaling is a challenging problem and that there is no single upscaling method that is unequivocally better than others. This is one motivation for the big attention so-called multiscale methods [95, 195] have received in recent years. These methods offer a promise of a more systematic and consistent way of bringing the impact of small-scale heterogeneity variations into simulations on a coarser scale.

What is the best upscaling method for a specific project will depend on many factors. The most important factor is probably what you intend to use the upscaled model for. If you are upscaling for a specific scenario, I strongly recommended that you use a method employing specific global information as this generally will enable you to perform more aggressive coarsening. If you, on the other hand, want to use the upscaled model to simulate a large variety of flow scenarios, or some yet unspecified scenario, you should pick a technique that is as robust as possible. Which technique to pick will depend on the type of heterogeneity you are facing, the flow patterns that exist in a reservoir, and so on. For relatively homogeneous reservoirs with flow patterns that mainly follow the axial directions, a simple averaging technique may be sufficient, whereas techniques that utilize some kind of global information are recommended for highly heterogeneous reservoirs with strong contrasts and large variations in correlation lengths. Other determining factors include the time and the computer resources you have available for upscaling, the number of petrophysical realizations you need to upscale, the required upscaling factor, and so on.

Regardless of your situation, I generally recommended that you try different upscaling and coarsening methods. Start by simple averaging methods that have a low computational cost, since these not only give you a first quick estimate of the upscaling uncertainty (or more generally the uncertainty in the petrophysical properties), but will also provide upper and lower bounds for more sophisticated upscaling methods. Check to see if there are features in your model that need to be resolved by adapting the coarse grid, and ensure that you do not create a coarse grid that gives problems for subsequent flow simulations; see e.g., the discussion in Section 14.7. Then you can gradually move towards more sophisticated upscaling methods. For local methods, you should try both sealing and linear pressure boundaries to provide lower and upper bounds. You may also need to check the use of oversampling methods to lessen the impact of boundary conditions used for localization.

At all stages, you should use representative single-phase simulations to validate your upscaled model against the original fine-scale model. To this end, you should compare flow fields computed both with TPFA and a consistent method to get an idea of the amount of grid-orientation effects. Likewise, I strongly recommend that you use various kinds of flow diagnostics like the well-allocation factors discussed in the examples in this chapter, partition functions that give you the volumetric connections between inflow and outflow, time-of-flight that gives time lines for displacement fronts, and so on. Herein, we have computed these quantities using finite-volume discretizations that are available in MRST, but these flow diagnostic measures can also be computed by streamline simulation [79], which is generally a very efficient tool for comparing geomodels and upscaled simulation models. Whether you choose one or the other is not that important. What is important is that you stick to using single-phase flow physics to avoid mixing in the effect of multiphase flow parameters, such as variations in relative permeabilities and capillary pressures, which need to be upscaled by other means.