Flow-based Upscaling of Reservoir Models

Knut–Andreas Lie

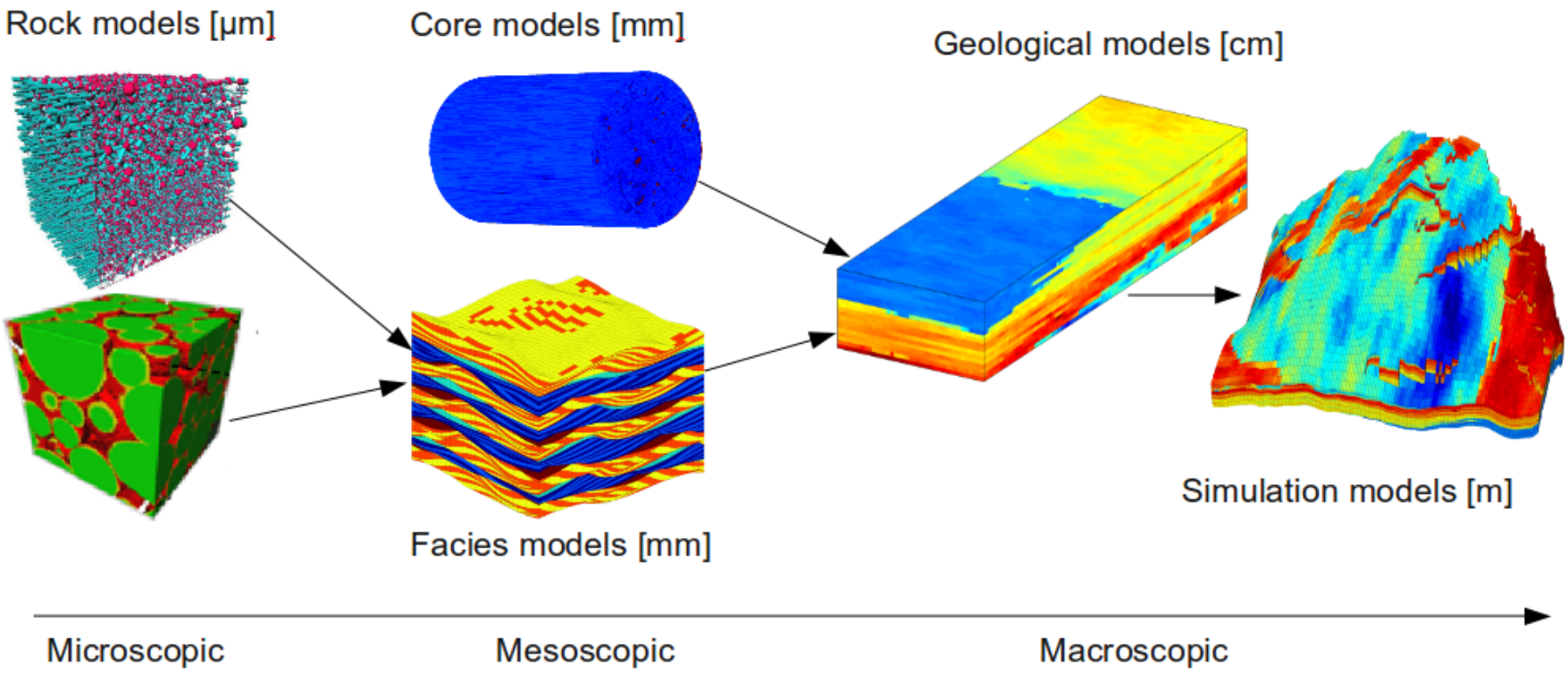
Department of Mathematics and Cybernetics, SINTEF Digital/
Department of Mathematical Sciences, NTNU, Norway

Multiscale Methods Summer School
June 26–29, 2017, Hasselt, Belgium

Outline

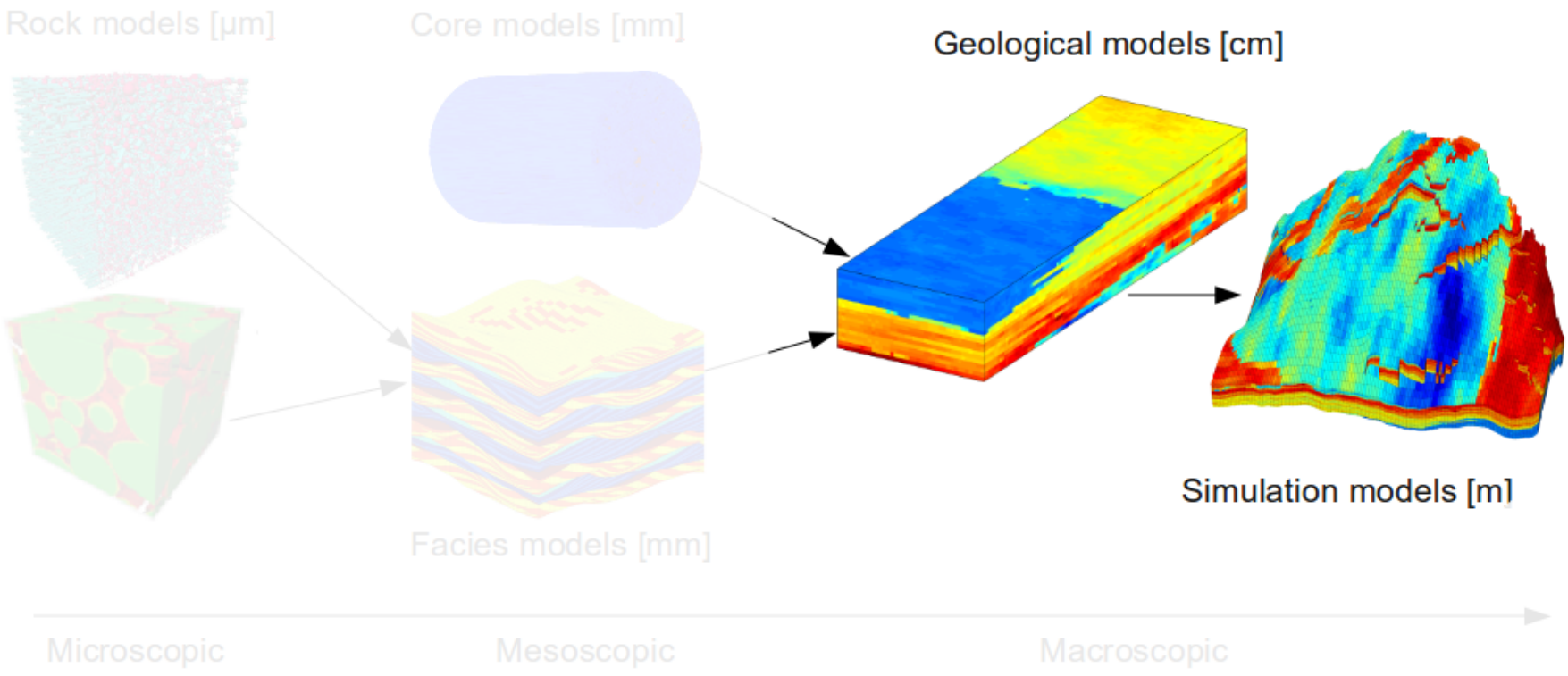
Quick recap: subsurface modeling

Multiscale structure induces a hierarchy of models



Quick recap: subsurface modeling

Multiscale structure induces a hierarchy of models

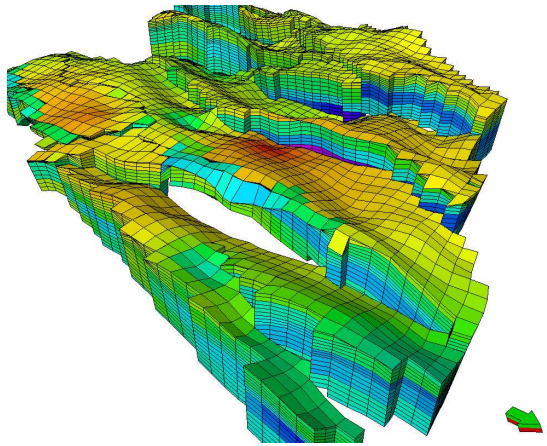


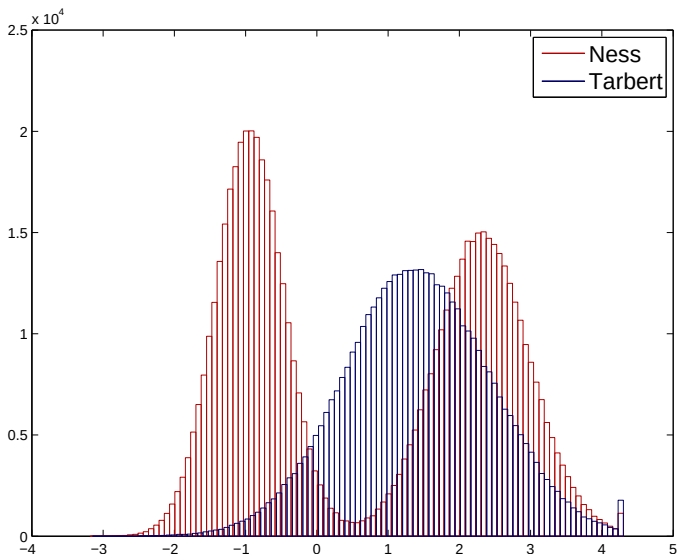
Quick recap: geological models

Articulation of the geologists' perception of the reservoir

Geological models:

- here: geo-cellular models
- describe the reservoir geometry: horizons, faults, etc
- typically generated using geostatistics
- give rock properties: permeability, porosity, net-to-gross





Rock properties:

- have a multiscale structure
- details on all scales impact flow
- permeability spans many orders of magnitude

Quick recap: geological models

Ever increasing complexity

As computational power increases, methods improve, and more data becomes available:

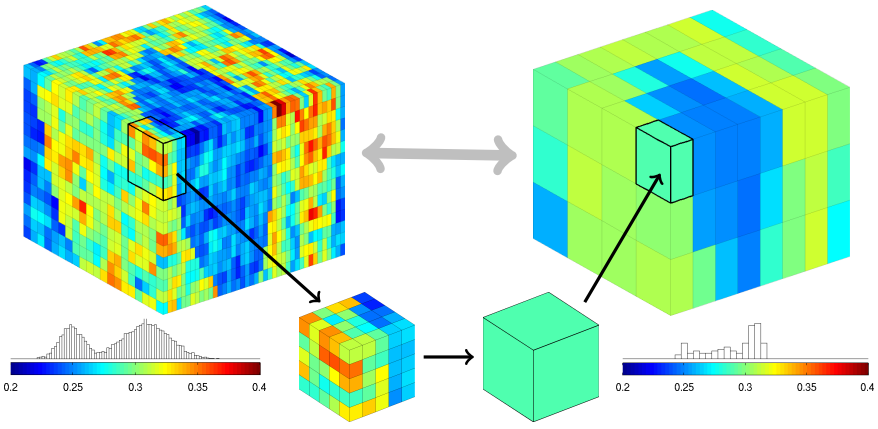
- Geological models increase in detail – today, geologists are routinely building models with 1–100 million cells
- A lot of the 'easy oil' is recovered $\longrightarrow$ models must account for fractures, faults, thin zones, high-contrast media, horizontal and deviated multilateral wells $\longrightarrow$ complex unstructured grid models

Building a coarse-scale model

Upscaling: geological $\longrightarrow$ simulation model

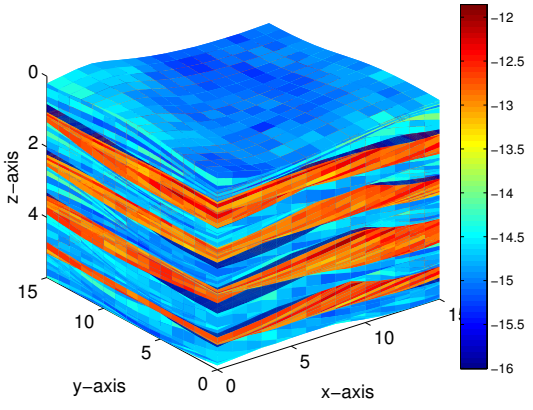
Single-phase upscaling:

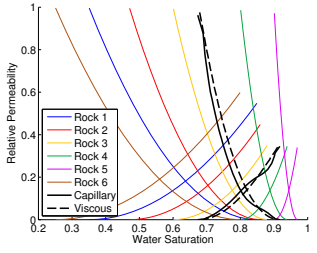
- Analytical methods: geometric, harmonic, arithmetic
- Flow-based: linear pressure, periodic, generic, specific, local, global, local-global, ...
- Wavelet, multi-resolution, renormalization, ...
- Ensemble methods

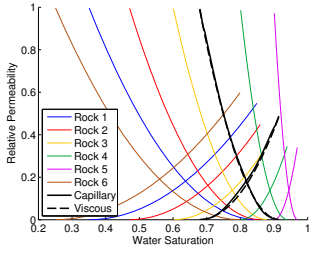


Two-phase/multiphase upscaling:

- Pseudo methods
- Steady-state upscaling: capillary limit, viscous limit, general steady-state
- Dynamic pseudo
- Vertical-equilibrium







Theoretical background: homogenization

Periodic medium with permeability $\mathbf{K}(\frac{x}{\varepsilon})$. Elliptic flow problem:

$$-\nabla \cdot \mathbf{K}(\frac{x}{\varepsilon})\nabla p_\varepsilon = q, \qquad \vec{v}_\varepsilon = -\mathbf{K}(\frac{x}{\varepsilon})\nabla p_\varepsilon$$

Theorem: There exists a constant symmetric and positive-definite tensor $\mathbf{K}_0$ such that $p_\varepsilon, \vec{v}_\varepsilon$ converge uniformly as $\varepsilon \to 0$ to the corresponding solutions of a homogenized equation

$$-\nabla \cdot \mathbf{K}_0 \nabla p_0 = q, \qquad \vec{v}_0 = -\mathbf{K}(\frac{x}{0})\nabla p_0$$

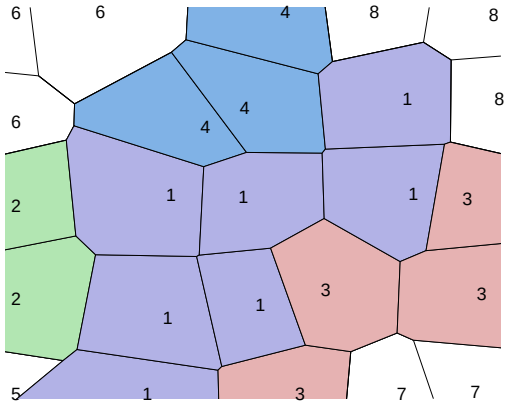
Unfortunately, natural rocks are seldom periodic...

Outline

Grid coarsening in MRST

In MRST, the 'coarse grid' concept always refers to a grid that is defined by partitioning another grid, which is referred to as the 'fine' grid:

- each cell in the fine grid `G` belongs to one, and only one, block in the coarse grid `CG`
- each block in `CG` consists of a *singly connected* subset of cells from `G`
- `CG` is defined by a partition vector p defined such that $p(i) = \ell$ if cell i in `G` belongs to block ℓ in `CG`



Tools for partitioning and coarsening of grids are found in two different modules of MRST:

- The `coarsegrid` module defines grid structure for representing coarse grids, and supplies routines for partitioning logically Cartesian grids
- The `agglom` module offers methods for defining coarse grids that adapt to geological features, flow patterns, etc

The coarsegrid module

Example: partition a Cartesian grid

Make the fine 4×4 grid:

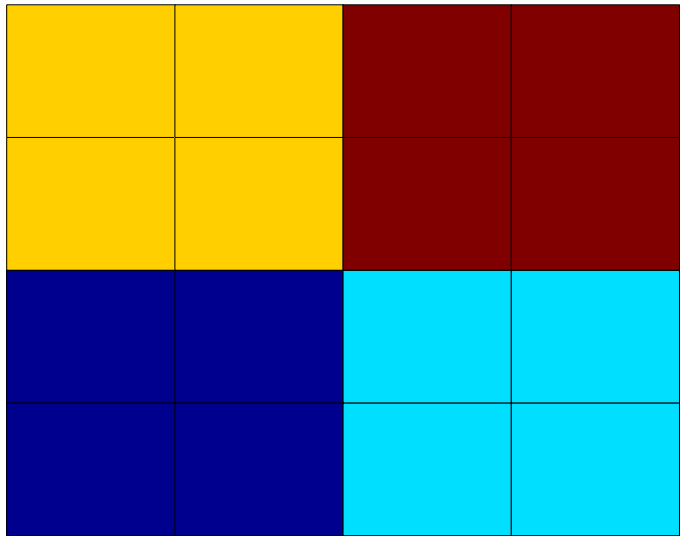
```
G = computeGeometry(cartGrid([4,4]));
```

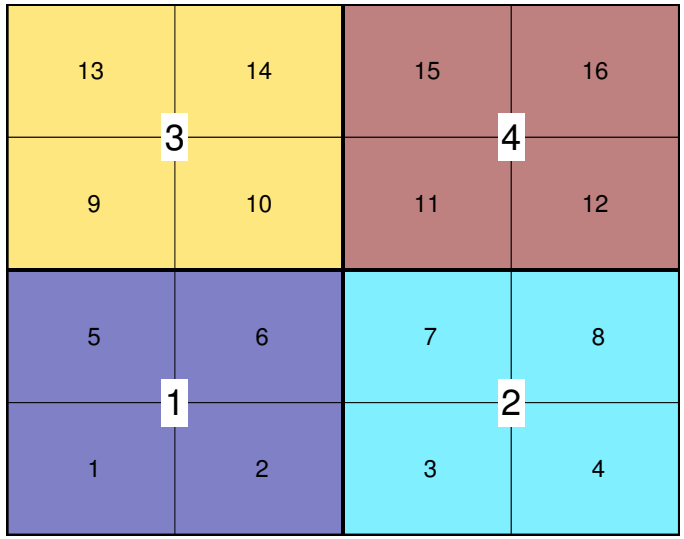
Partition into a 2×2 grid:

```
p = partitionCartGrid(G.cartDims, [2,2]);
plotCellData(G,p,'EdgeColor','w','EdgeAlpha',.2)
```

Construct a coarse grid:

```
CG = generateCoarseGrid(G, p);
CG = coarsenGeometry(CG);
```





The coarsegrid module

Example: partition a faulted model

Make the fine grid:

```
nx = 20; ny = 20; nz = 8;
Nx =  5; Ny =  4; Nz = 2;
grdecl = simpleGrdecl([nx, ny, nz], 0.15);
G = processGRDECL(grdecl);
G = computeGeometry(G);
plotGrid(G); view(30,20); axis off
```

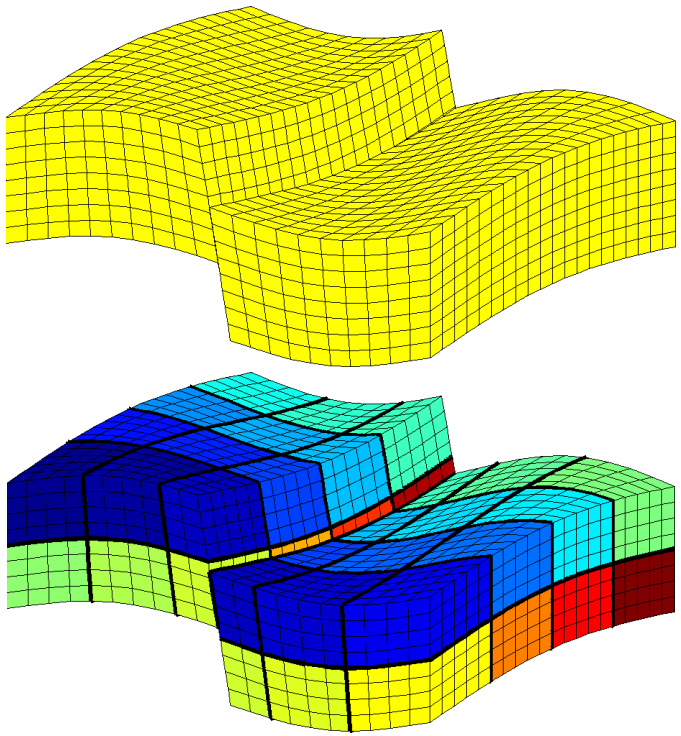
Next, we introduce a uniform partition (or in general, a load-balanced linear partition):

```
p = partitionUI(G,[Nx, Ny, Nz]);
plotCellData(G, p, 'EdgeColor', 'k');
outlineCoarseGrid(G, p, ...
      'FaceColor','none','LineWidth',3);
```

Then, we split blocks that are not connected and construct a coarse grid:

```
p  = processPartition(G, p);
CG = generateCoarseGrid(G, p)
```

in which the fields have the same structure and interpretation as in standard grids



```
CG =

        cells: [1x1 struct]
        faces: [1x1 struct]
    partition: [3200x1 double]
       parent: [1x1 struct]
      griddim: 3
         type: {'generateCoarseGrid'}
```

Outline

Upscaling porosity

Porosity is an additive property and can be upscaled through a standard volume average:

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

Upscaling porosity

Porosity is an additive property and can be upscaled through a standard volume average:

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

In MRST: assume `q` is a partition vector on grid `G`

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

Here, we use `max(..,eps)` to safeguard against division by zero if `q` is not contiguous or all cells in a block have zero porosity

Upscaling additive properties

For other additive properties, we weight by pore volume:

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

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

Upscaling additive properties

For other additive properties, we weight by pore volume:

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

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

Let us verify that this is the correct average:

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

Upscaling additive properties

For other additive properties, we weight by pore volume:

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

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

Let us verify that this is the correct average:

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

Other quantities like concentration should be weighted by saturation.

For integer properties like rock type, one should use a majority vote

Upscaling absolute permeability

Single-phase flow:

$$-\nabla \cdot \mathbf{K} \nabla p = f, \qquad \text{in } \Omega$$

Subdivide grid into coarse blocks B,
seek tensors $\mathbf{K}^*$ such that

$$\int_B \mathbf{K} \nabla p \, dx = \mathbf{K}^* \int_B \nabla p \, dx,$$

Upscaling absolute permeability

Single-phase flow:

$$-\nabla \cdot \mathbf{K} \nabla p = f, \qquad \text{in } \Omega$$

Subdivide grid into coarse blocks B, seek tensors $\mathbf{K}^*$ such that

$$\int_B \mathbf{K} \nabla p \, dx = \mathbf{K}^* \int_B \nabla p \, dx,$$

Use Darcy's law on the coarse scale

$$\bar{u} = -\mathbf{K}^* \overline{\nabla p}$$

net flow rate through B

average pressure gradient inside B

Upscaling absolute permeability

Single-phase flow:

$$-\nabla \cdot \mathbf{K} \nabla p = f, \qquad \text{in } \Omega$$

Subdivide grid into coarse blocks B, seek tensors $\mathbf{K}^*$ such that

$$\int_B \mathbf{K} \nabla p \, dx = \mathbf{K}^* \int_B \nabla p \, dx,$$

Use Darcy's law on the coarse scale

$$\bar{u} = -\mathbf{K}^* \overline{\nabla p}$$

net flow rate through B

average pressure gradient inside B

Observations:

- $\mathbf{K}^*$ is not uniquely defined by the second equation
- or conversely, there does not exist a unique $\mathbf{K}^*$ so that this equation holds for all pressures
- $\mathbf{K}^*$ depends on flow through B and hence on the boundary conditions on ∂B
- there is no guarantee that $\vec{z} \cdot \mathbf{K}^* \vec{z} > 0$ for all nonzero $\vec{z}$. Hence, the upscaling problem is fundamentally ill-posed

Averaging absolute permeability

Simplest approach – the power average:

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

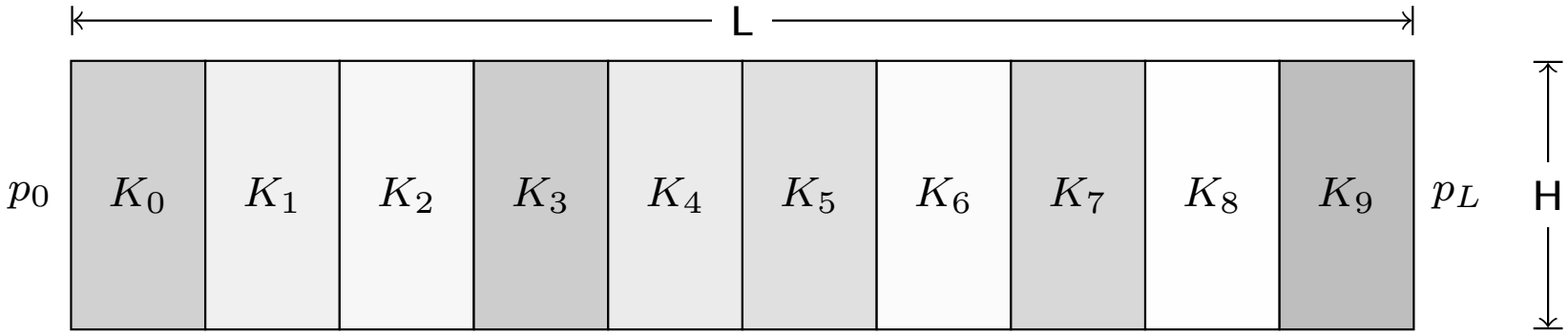
$p = 1$: arithmetic mean $\qquad$ $p = -1$: harmonic mean

Geometric mean: obtained in the limit $p \to 0$ as

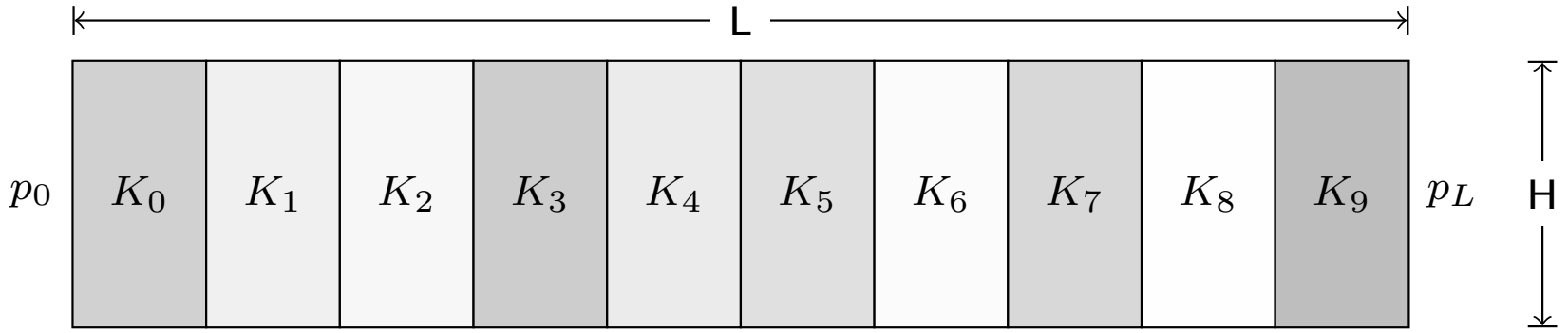
$$\mathbf{K}^* = A_0(\mathbf{K}) = \exp\Big(\frac{1}{|\Omega|}\int_\Omega \log(\mathbf{K}(\vec{x}))\, d\vec{x}\Big)$$

Theoretical backing in the Wiener bounds: $A_{-1}(K) \leq \mathbf{K}^* \leq A_1(\mathbf{K})$

Motivation: perpendicular layers



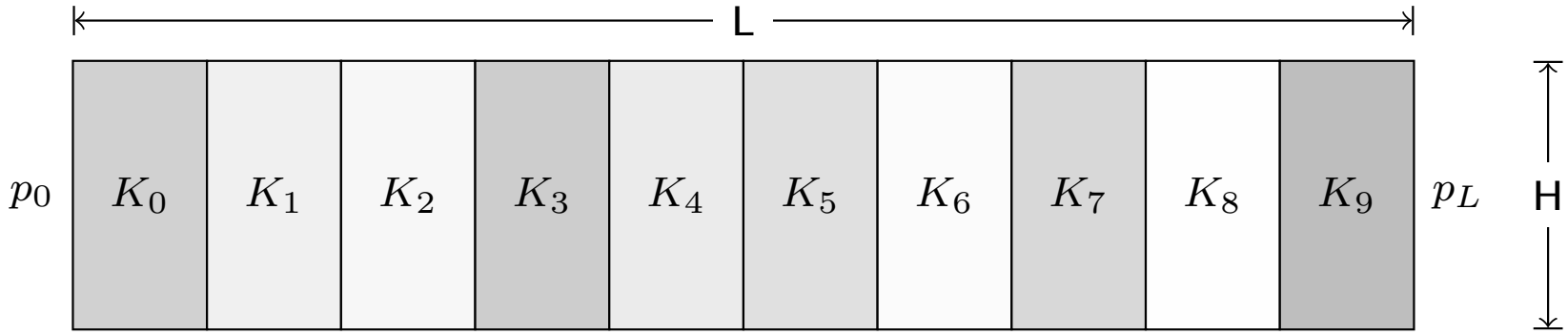
Motivation: perpendicular layers



From the flow equations:

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

Motivation: perpendicular layers



From the flow equations:

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

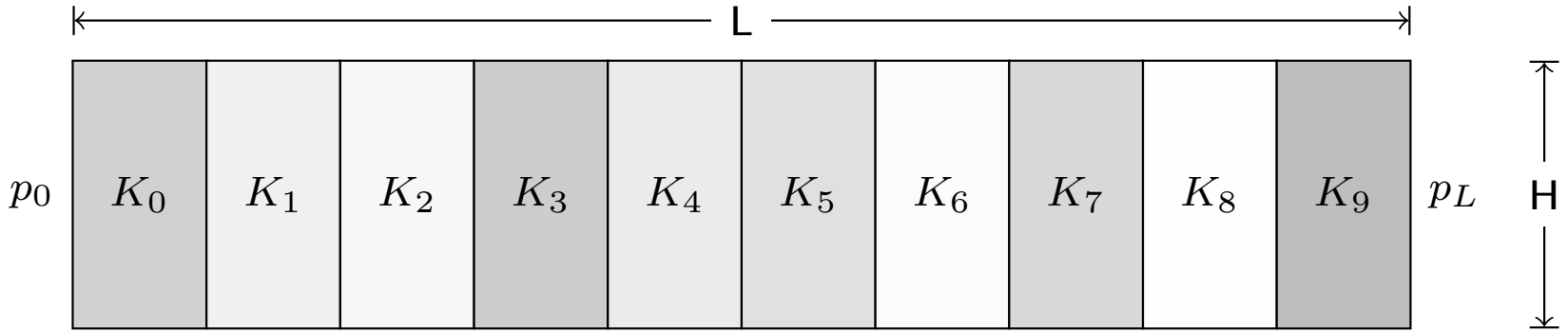
The constant v follows from Darcy's law on the coarse scale:

$$v = -K^*[p_L - p_0]/L$$

or can be determined from the definition of $\mathbf{K}^*$:

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

Motivation: perpendicular layers

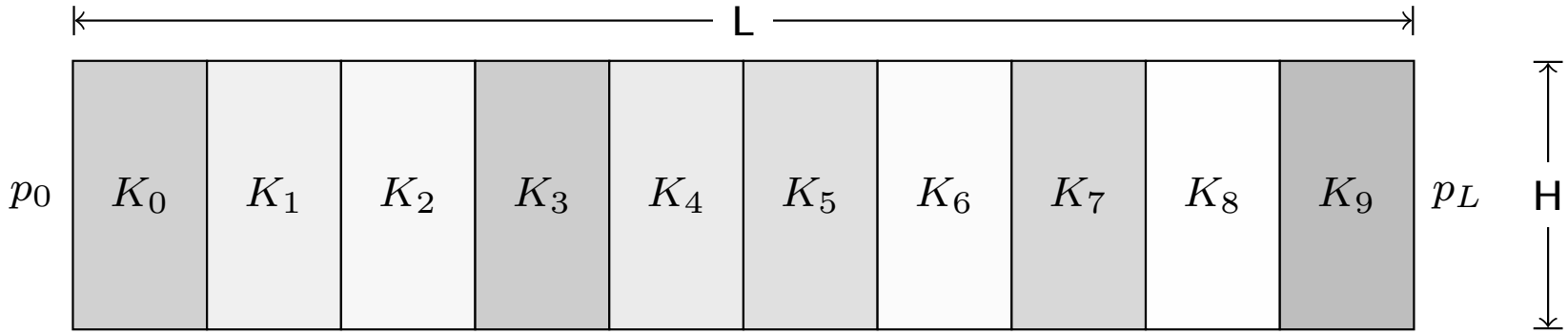


Starting from the average pressure gradient, we derive

$$\int_0^L p'(x)dx = -\int_0^L \frac{v}{K(x)}dx$$

$$= \underbrace{K^* \frac{p_L - p_0}{L}}_{v} \int_0^L \frac{1}{K(x)}dx = K^* \Big(\int_0^L p'(x)dx\Big)\Big(\frac{1}{L}\int_0^L \frac{1}{K(x)}dx\Big)$$

Motivation: perpendicular layers



Starting from the average pressure gradient, we derive

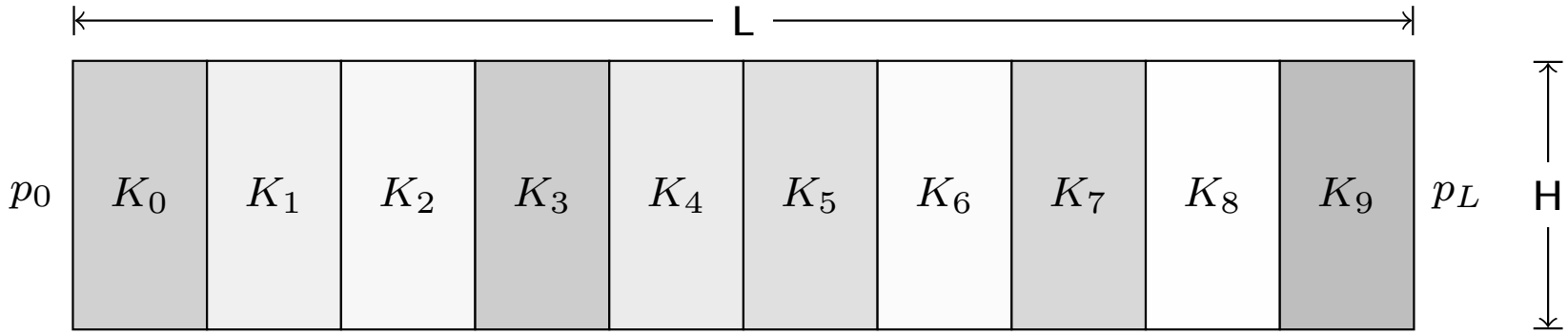
$$\int_0^L p'(x)dx = -\int_0^L \frac{v}{K(x)}dx$$

$$= \underbrace{K^* \frac{p_L - p_0}{L}}_{v} \int_0^L \frac{1}{K(x)}dx = K^* \Big(\int_0^L p'(x)dx\Big)\Big(\frac{1}{L}\int_0^L \frac{1}{K(x)}dx\Big)$$

from which it follows that

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

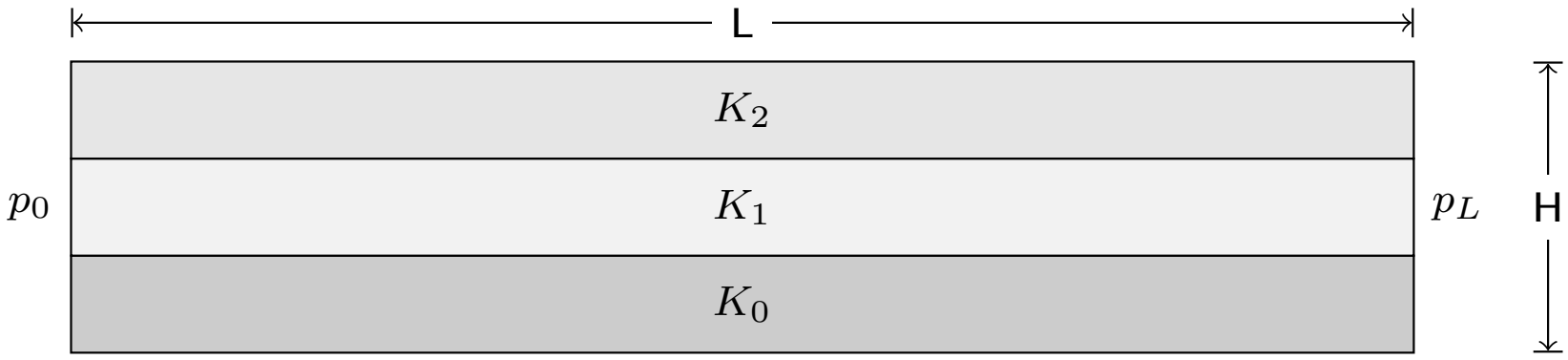
Motivation: perpendicular layers



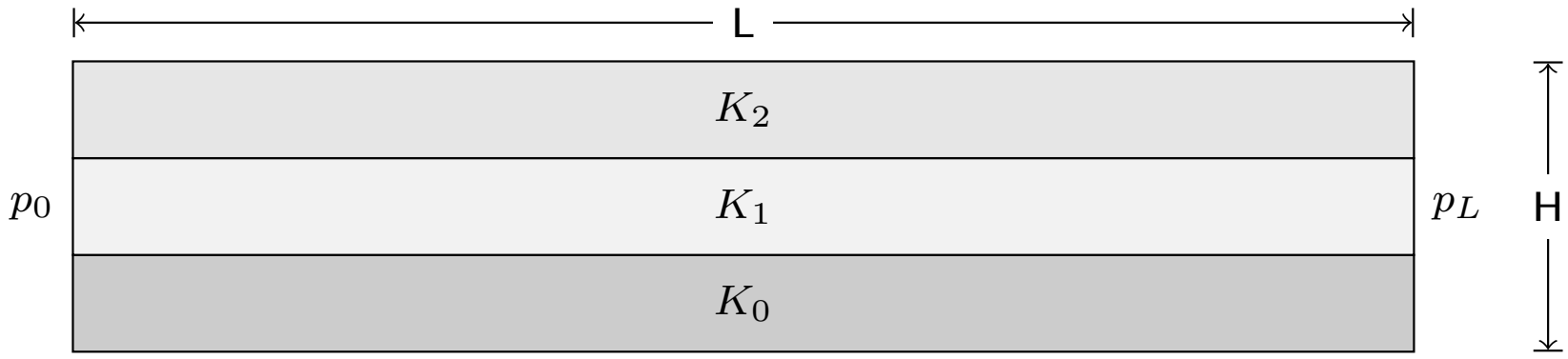
In MRST:

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

Motivation: perfectly stratified medium



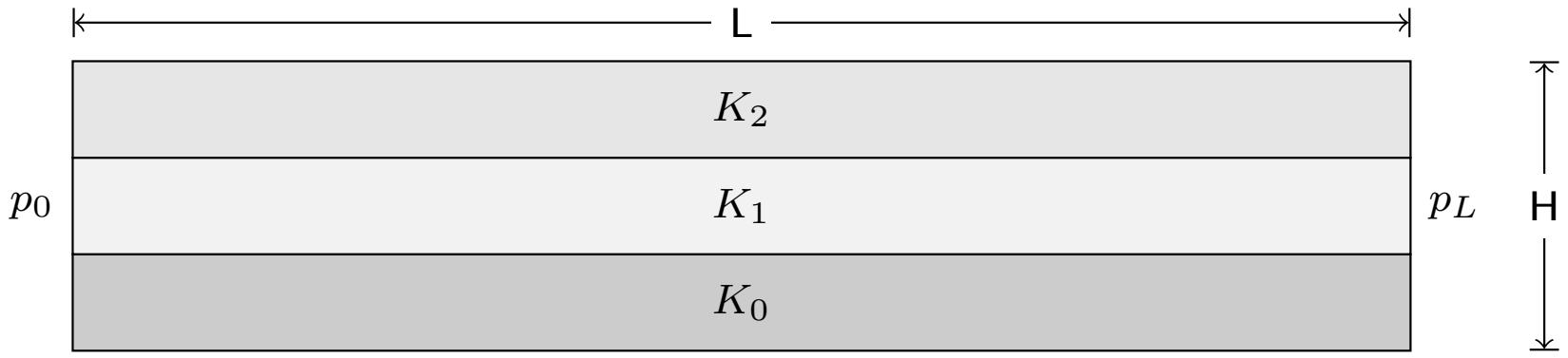
Motivation: perfectly stratified medium



Linear pressure drop in the x-direction:

$$p(x, y) = p_0 + x(p_L - p_0)/L$$

Motivation: perfectly stratified medium



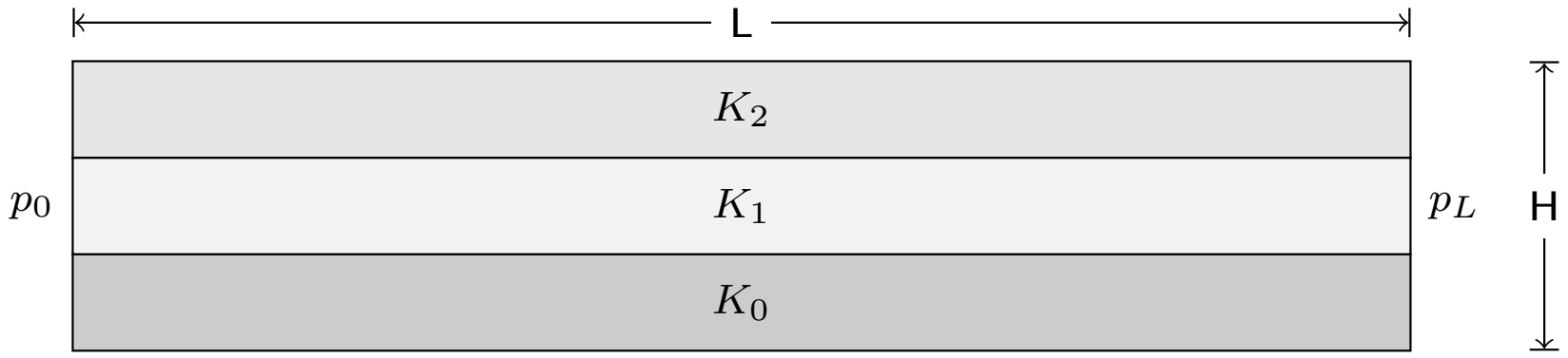
Linear pressure drop in the x-direction:

$$p(x, y) = p_0 + x(p_L - p_0)/L$$

Using definition of $\mathbf{K}^*$:

$$K^* \int_0^H \int_0^L \partial_x p(x, y)\, dx\, dy$$
$$= \int_0^H \int_0^L K(x, y) \partial_x p(x, y)\, dx\, dy$$

Motivation: perfectly stratified medium



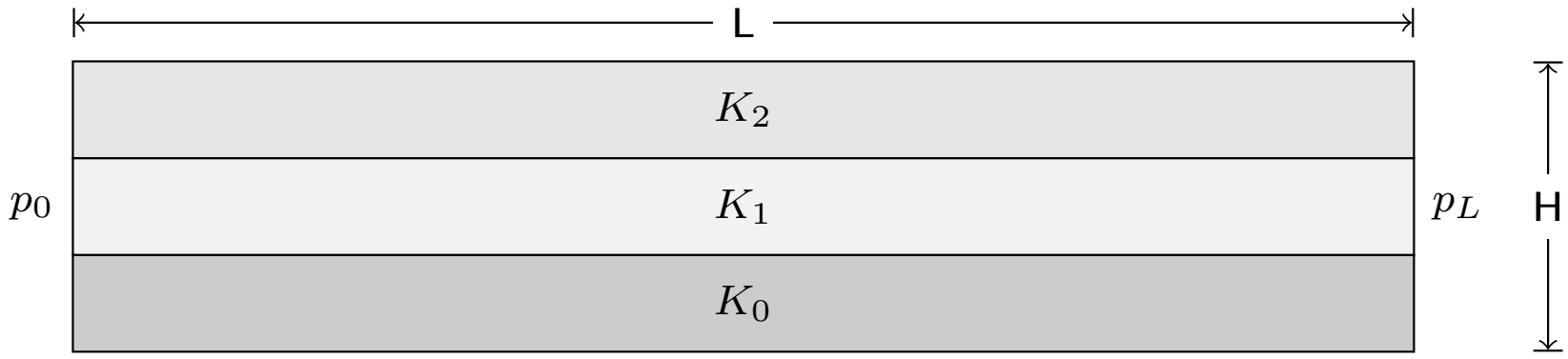
Linear pressure drop in the x-direction:

$$p(x, y) = p_0 + x(p_L - p_0)/L$$

Using definition of $\mathbf{K}^*$:

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

Motivation: perfectly stratified medium



Using definition of $\mathbf{K}^*$:

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

from which it follows that

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

Harmonic-arithmetic averaging

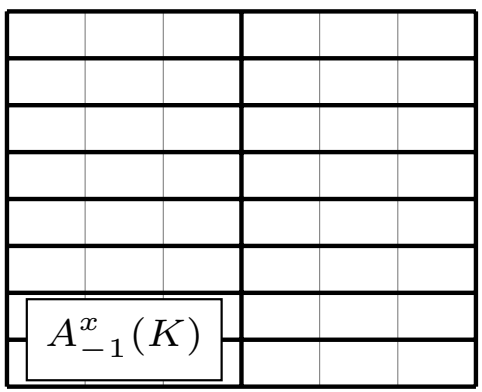
$$\mathbf{K}^* = \begin{bmatrix} A_1^{yz}(A_{-1}^{x}(\mathbf{K})) & 0 & 0 \\ 0 & A_1^{xz}(A_{-1}^{y}(\mathbf{K})) & 0 \\ 0 & 0 & A_1^{xy}(A_{-1}^{z}(\mathbf{K})) \end{bmatrix}$$

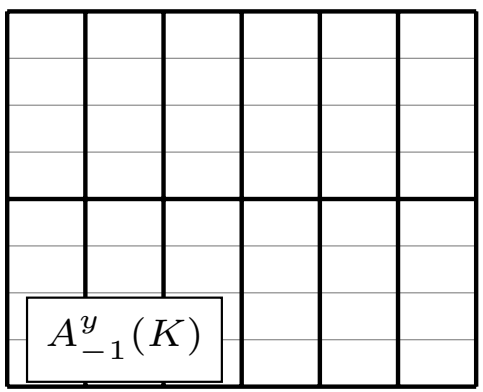
Harmonic-arithmetic averaging

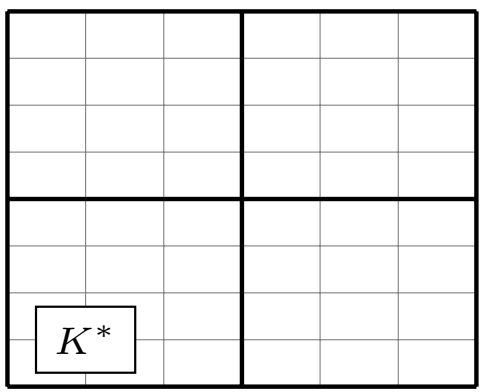
$$\mathbf{K}^* = \begin{bmatrix} A_1^{yz}(A_{-1}^x(\mathbf{K})) & 0 & 0 \\ 0 & A_1^{xz}(A_{-1}^y(\mathbf{K})) & 0 \\ 0 & 0 & A_1^{xy}(A_{-1}^z(\mathbf{K})) \end{bmatrix}$$

In MRST:

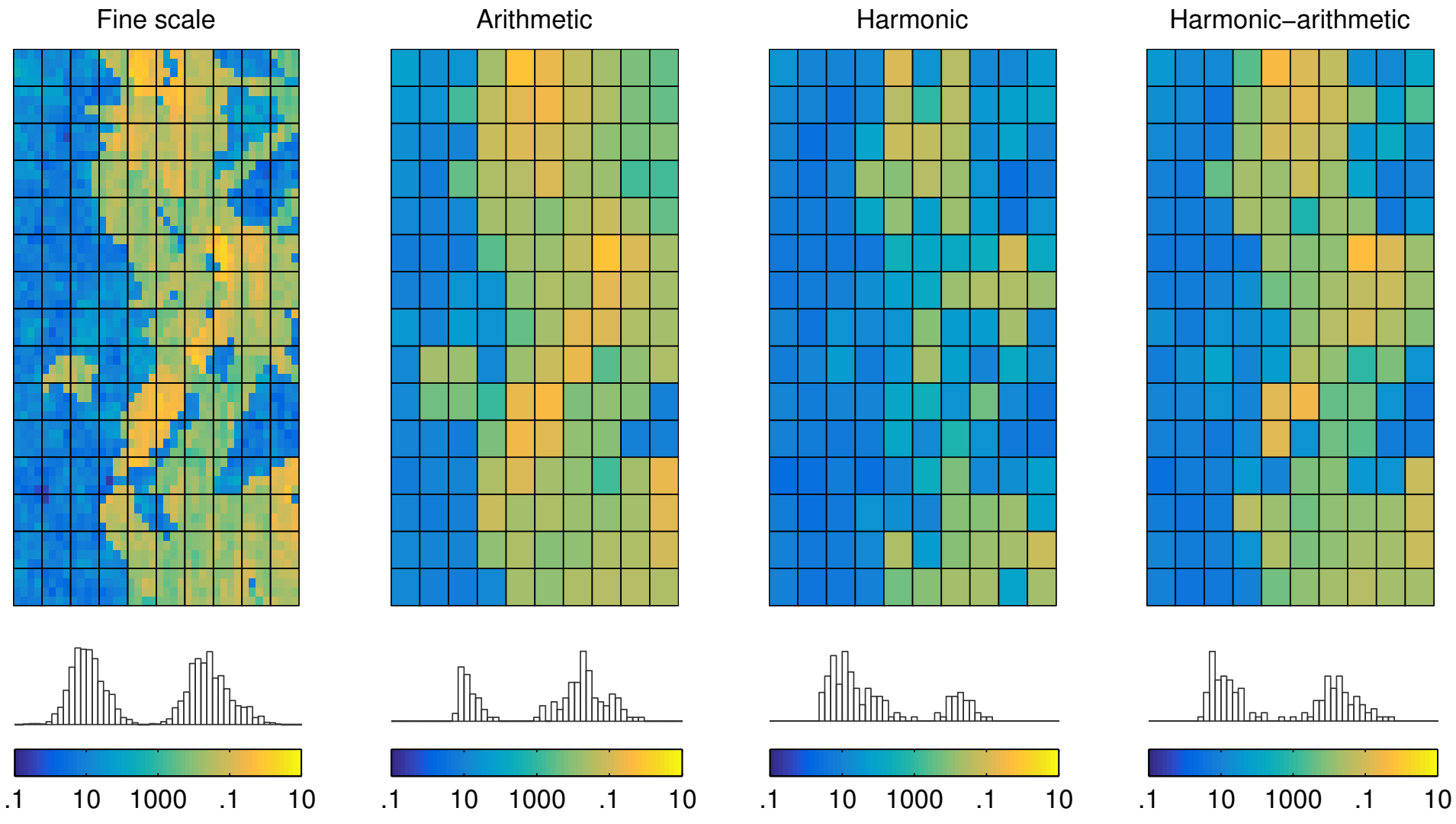
```
q = partitionUI(G, coarse);
vol = G.cells.volumes;
for i=1:size(rock.perm,2)
   dims = G.cartDims; dims(i)=coarse(i);
   qq = partitionUI(G, dims);
   K = accumarray(qq,vol)./accumarray(qq,vol./rock.perm(:,i));
   crock.perm(:,i) = accumarray(q,K(qq).*vol)./accumarray(q,vol);
end
```



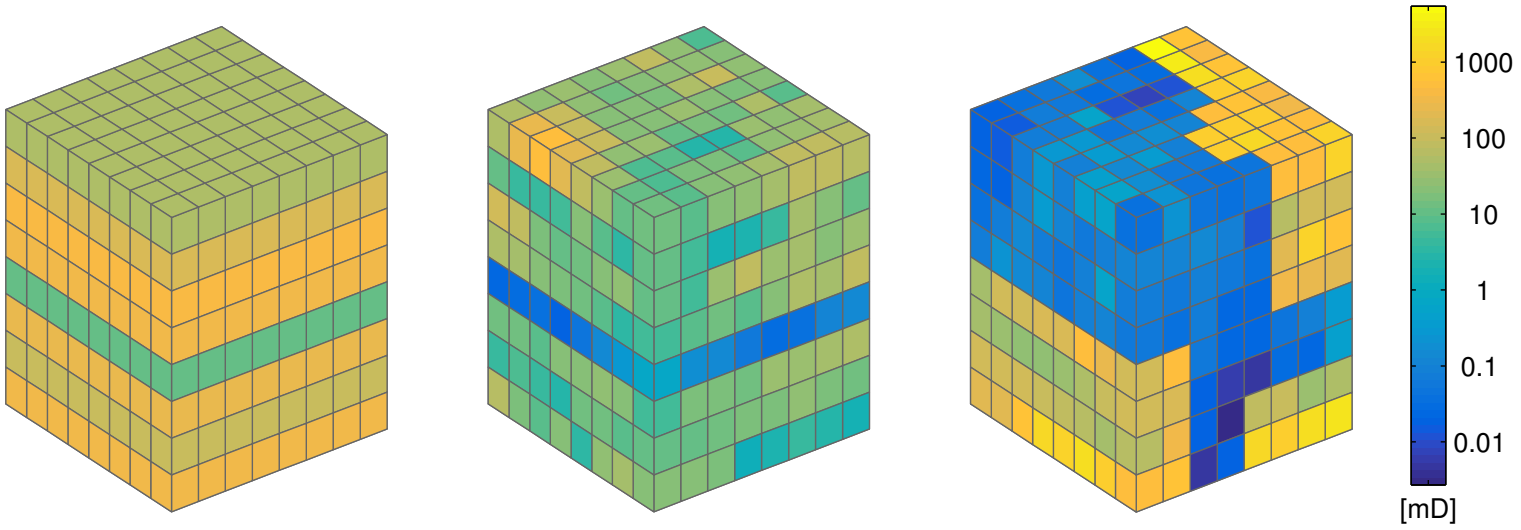




Example: Layer 46, SPE 10



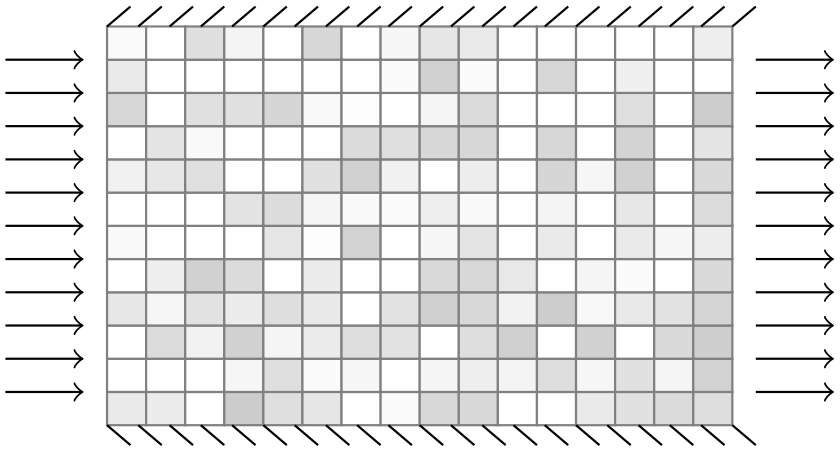
Example: ratio of fine and coarse-scale fluxes

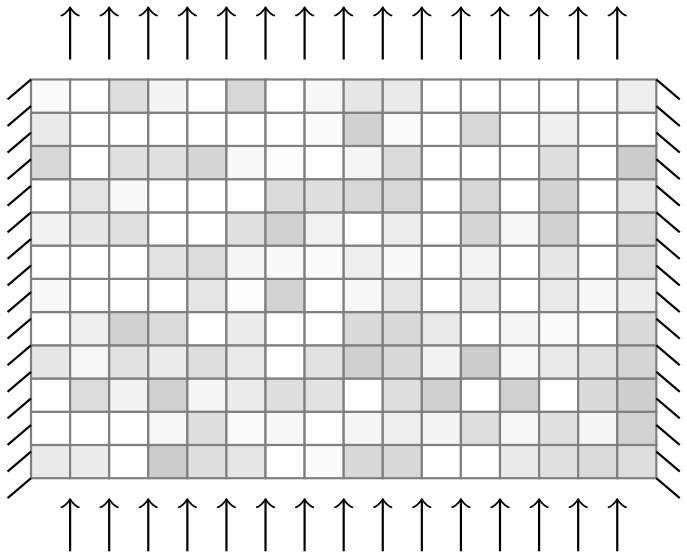


Model	Flow pattern	Arithmetic	Harmonic	Harm-arith
Layered	East → West	1.0000	0.2662	1.0000
	North → South	1.0000	0.2662	1.0000
	Top → Bottom	3.7573	1.0000	1.0000
Tarbert	East → West	1.6591	0.0246	0.8520
	North → South	1.6337	0.0243	0.8525
	Top → Bottom	47428.0684	0.3239	0.8588
Upper Ness	East → West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

Outline

Flow-based upscaling

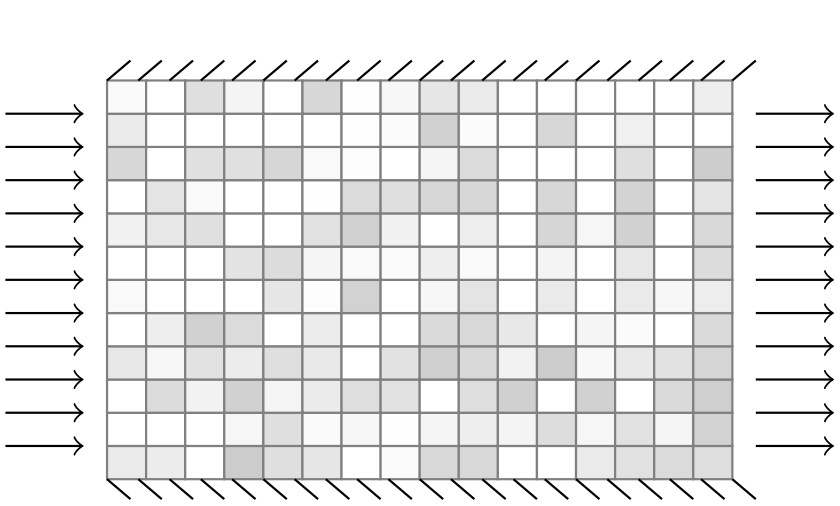


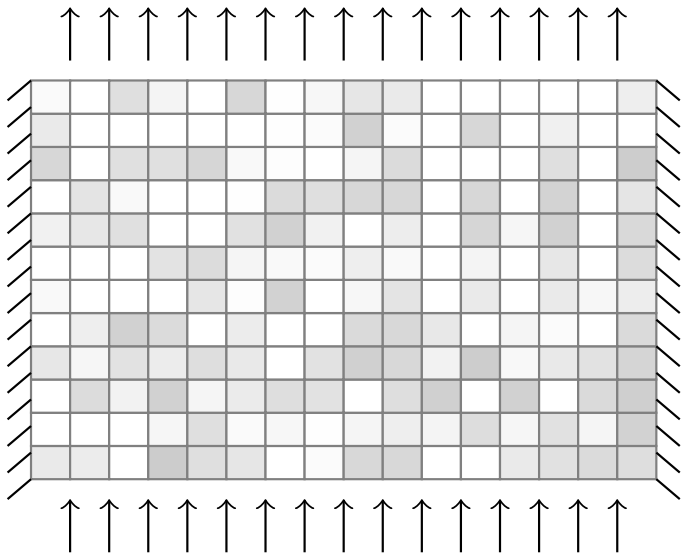


Emulate laboratory setup: pressure drop in each axial direction and sealing conditions along the other boundaries $\longrightarrow$ diagonal tensor:

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

Flow-based upscaling





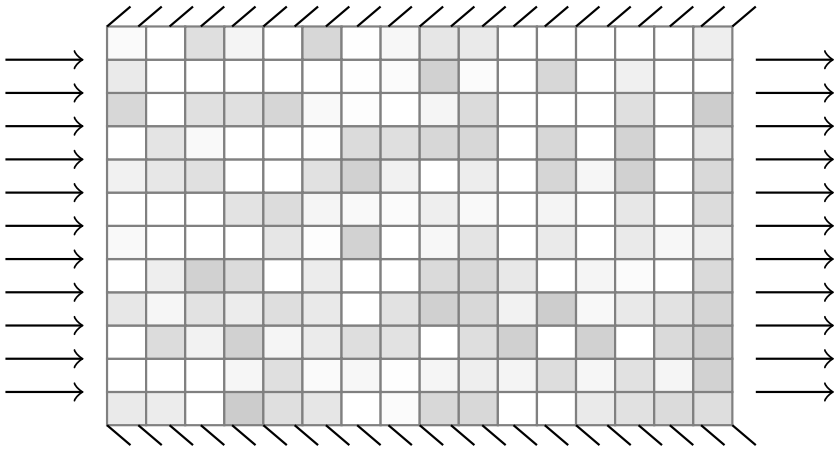
Emulate laboratory setup: pressure drop in each axial direction and sealing conditions along the other boundaries $\longrightarrow$ diagonal tensor:

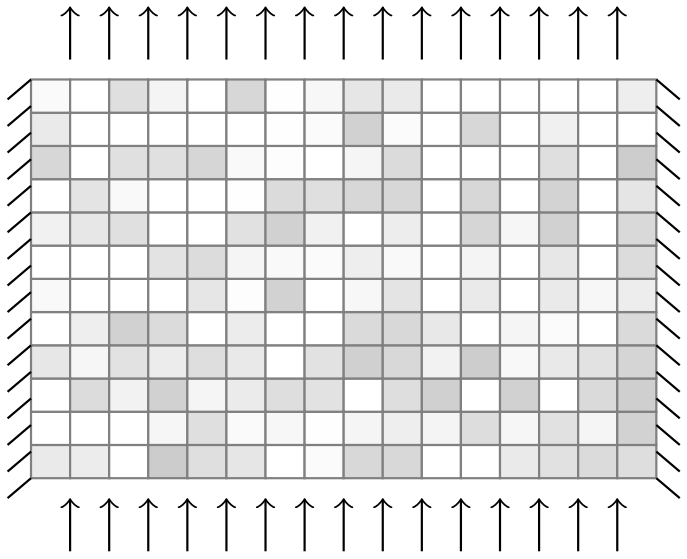
$$K_{xx} = -\frac{v_x L_x}{\Delta p_x}, \qquad K_{yy} = -\frac{v_y L_y}{\Delta p_y}, \qquad K_{zz} = -\frac{v_z L_z}{\Delta p_z},$$

Only valid when **K** is symmetric with respect to the faces of the coarse grid, i.e., the block is surrounded by mirror images of itself.

Tends to introduce bias toward lower permeability values: thickening shale barriers and narrowing sand channels.

Flow-based upscaling



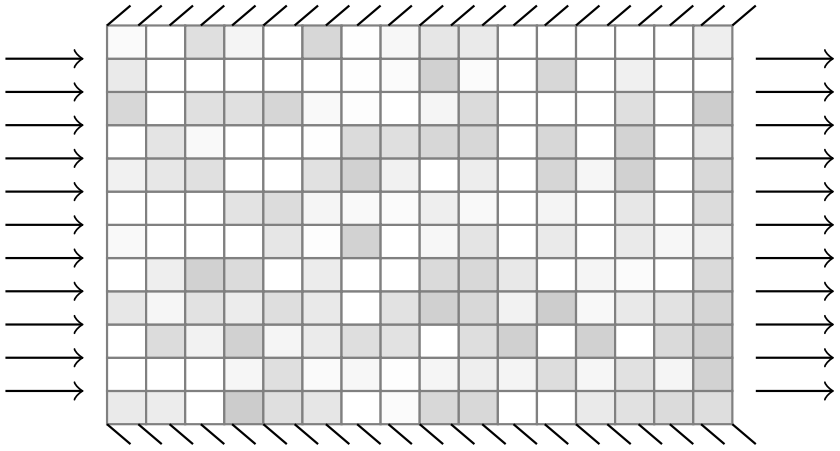


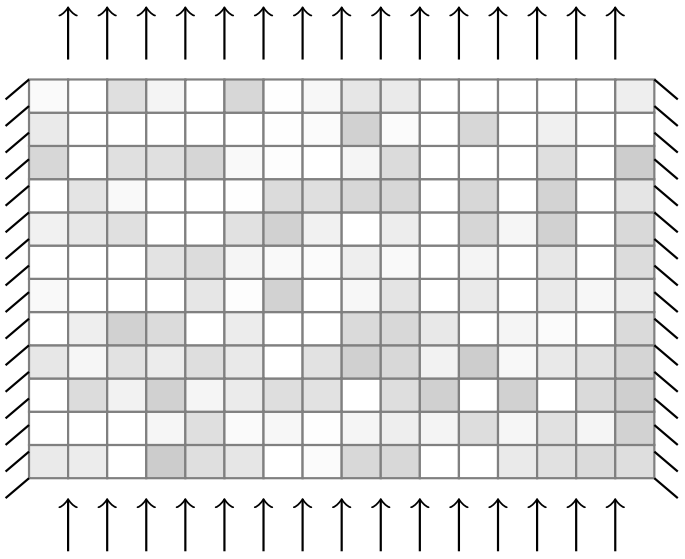
In MRST, for a rectangular grid:

```
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 `bcl` and `bcr` are templates storing all boundary conditions. Although not necessary now, we will use them later.

Flow-based upscaling

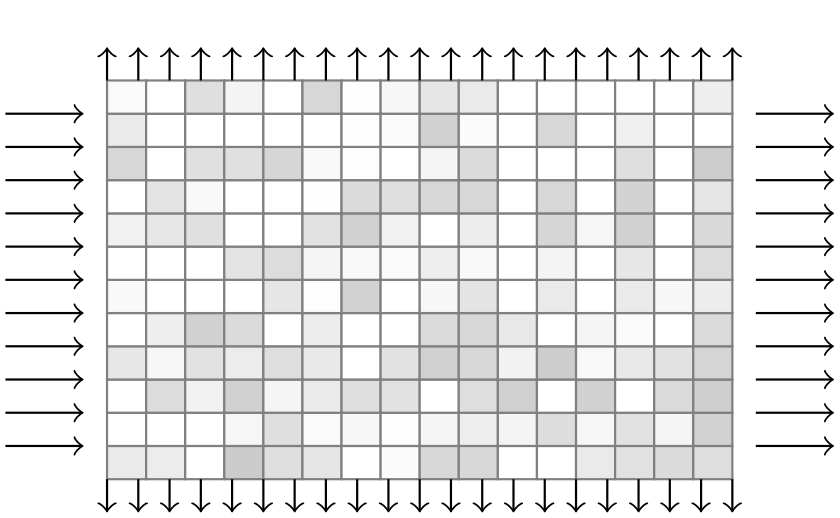


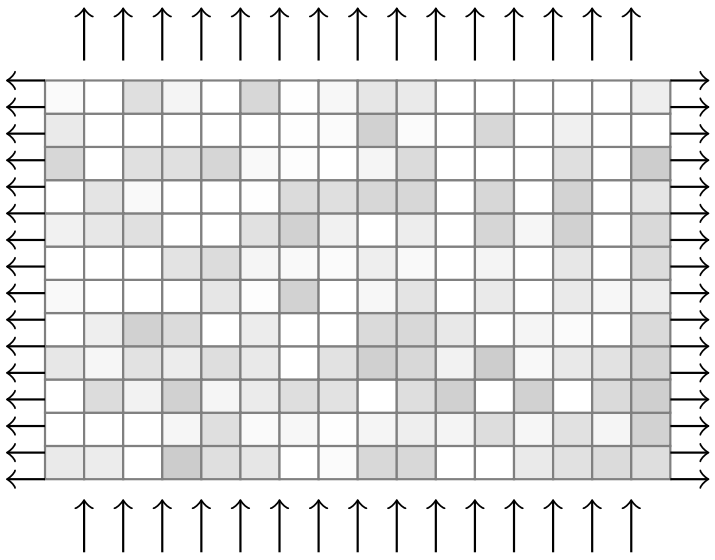


The upscaling can then be computed by looping over all axial directions:

```
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);
```

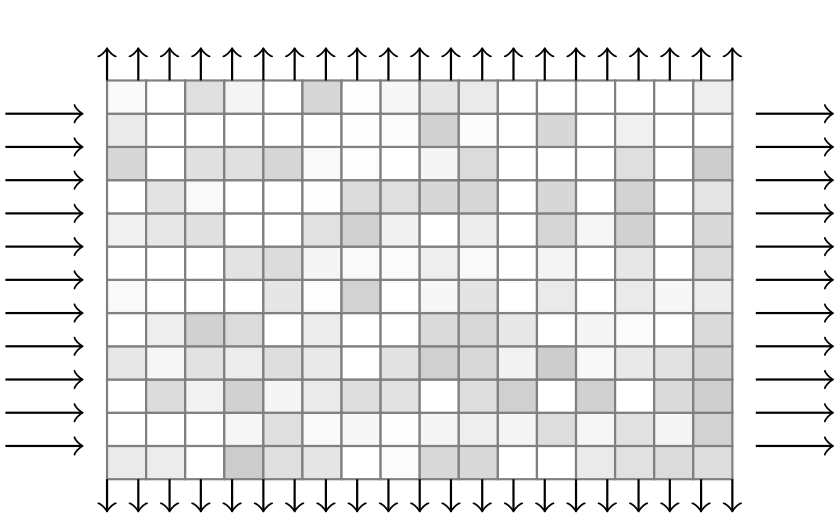
Flow-based upscaling

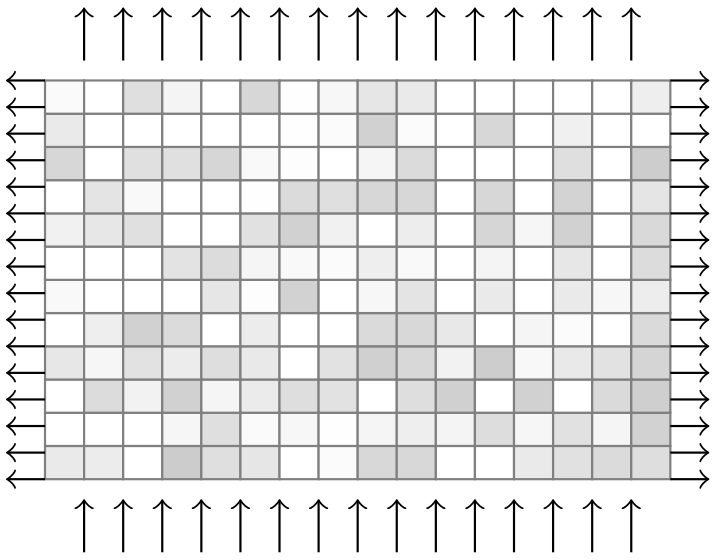




More natural: pressure drop in each axial direction, but open boundaries, or alternatively a linear pressure drop parallel to flow direction $\longrightarrow \mathbf{K}^*$ is full tensor.

Flow-based upscaling



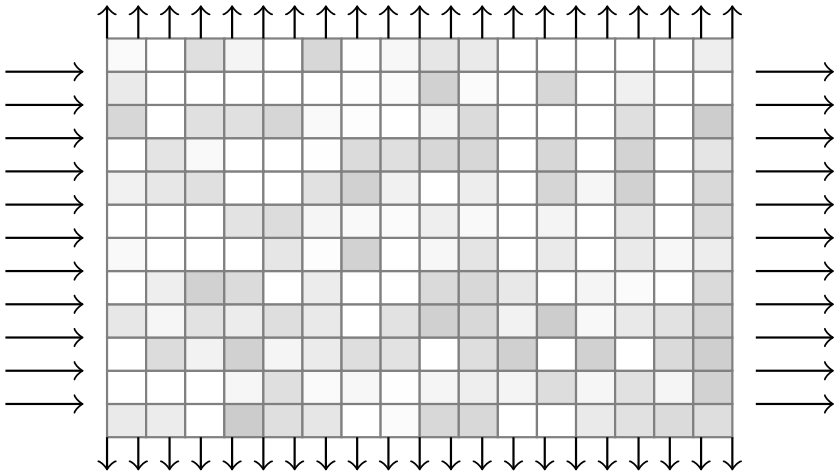


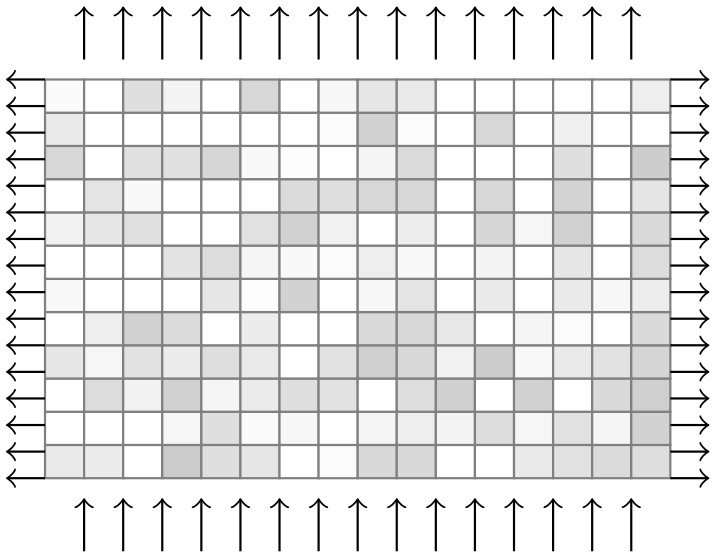
Another popular option: periodic boundary conditions, here for the x-component:

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

Gives symmetric and positive definite $\mathbf{K}^*$

Flow-based upscaling



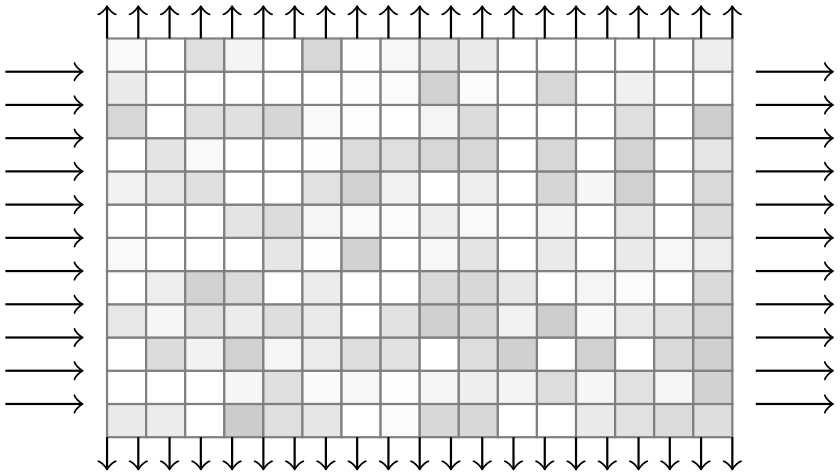


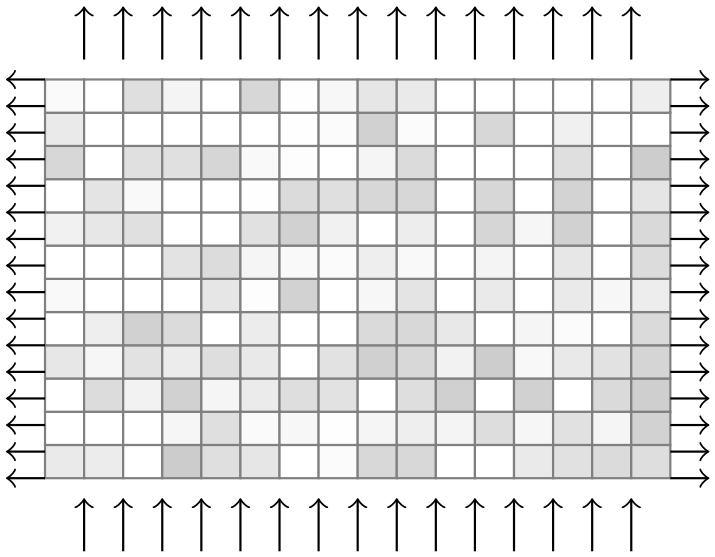
We start by modifying the grid structure

```
[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. The routine also sets up an appropriate structure for representing periodic boundary conditions.

Flow-based upscaling

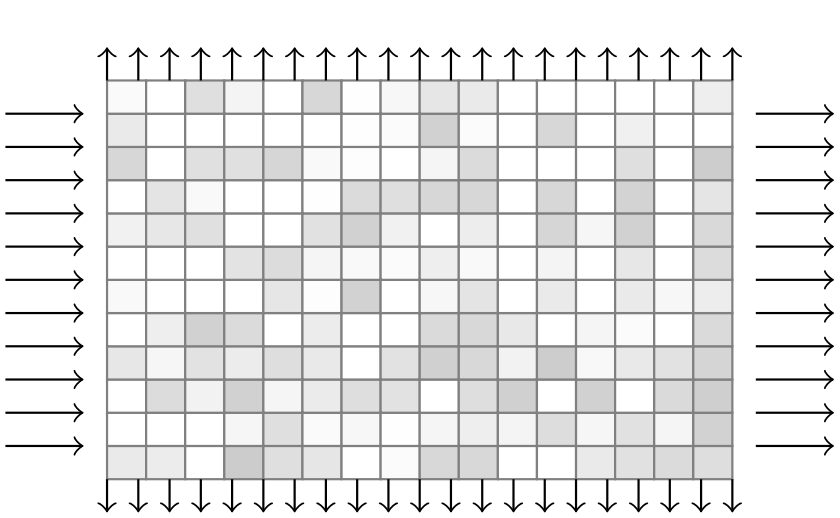


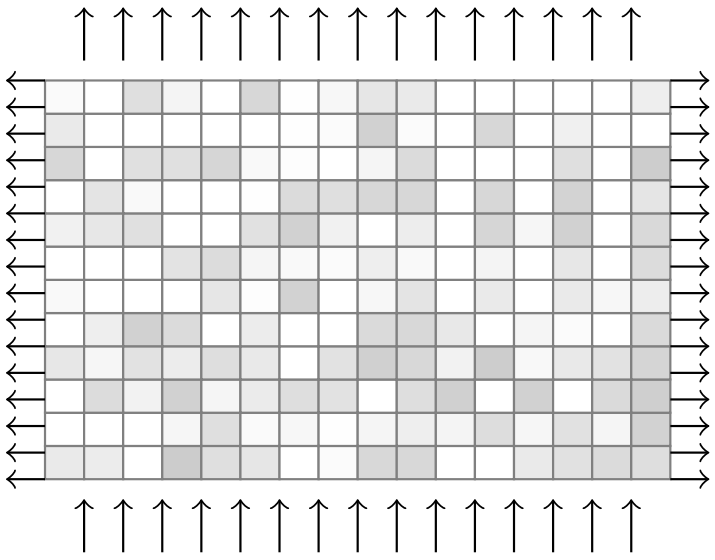


```
dp = Dp{1}*eye(d); nbcp = bcp;
for i=1:d
    for j=1:d, nbcp.value(bcp.tags==j)=dp(j,i); end
    xr = incompTPFA(initResSol(Gp, 100*barsa, 1), Gp, hT, fluid, 'bcp', nbcp);
    for j=1:d
        v(j,i) = sum(xr.flux(ofaces{j})) / sum(Gp.faces.areas(ofaces{j}));
    end
end
```

First inner loop: extracts the correct pressure drop to be included in the periodic boundary conditions from the diagonal matrix `dp`. Second loop: Compute total flux across all outflow faces

Flow-based upscaling



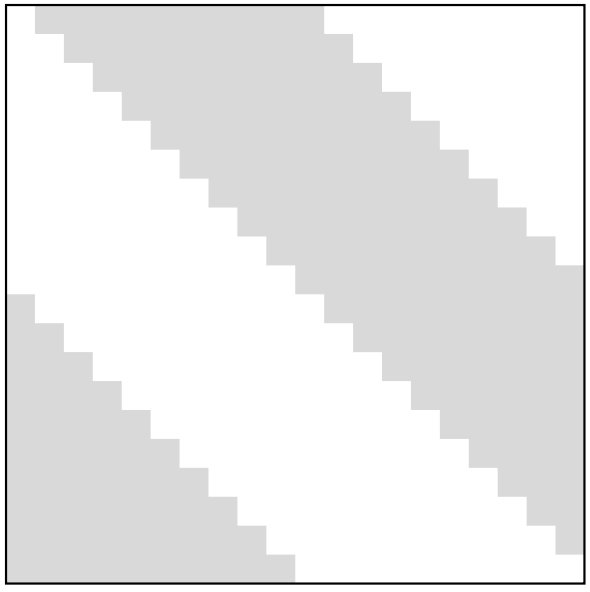


We can then compute the average pressure drop and invert Darcy's law to compute $\mathbf{K}^*$

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

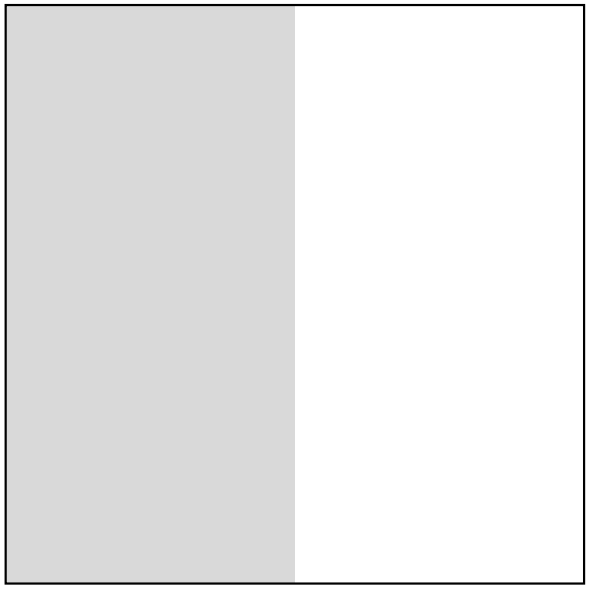
Sealing and periodic upscaling are implemented in two functions in the `upscaling` module: `upscalePermeabilityFixed`, `upscalePermeabilityPeriodic`

Example: two simple examples



10 mD

30 mD



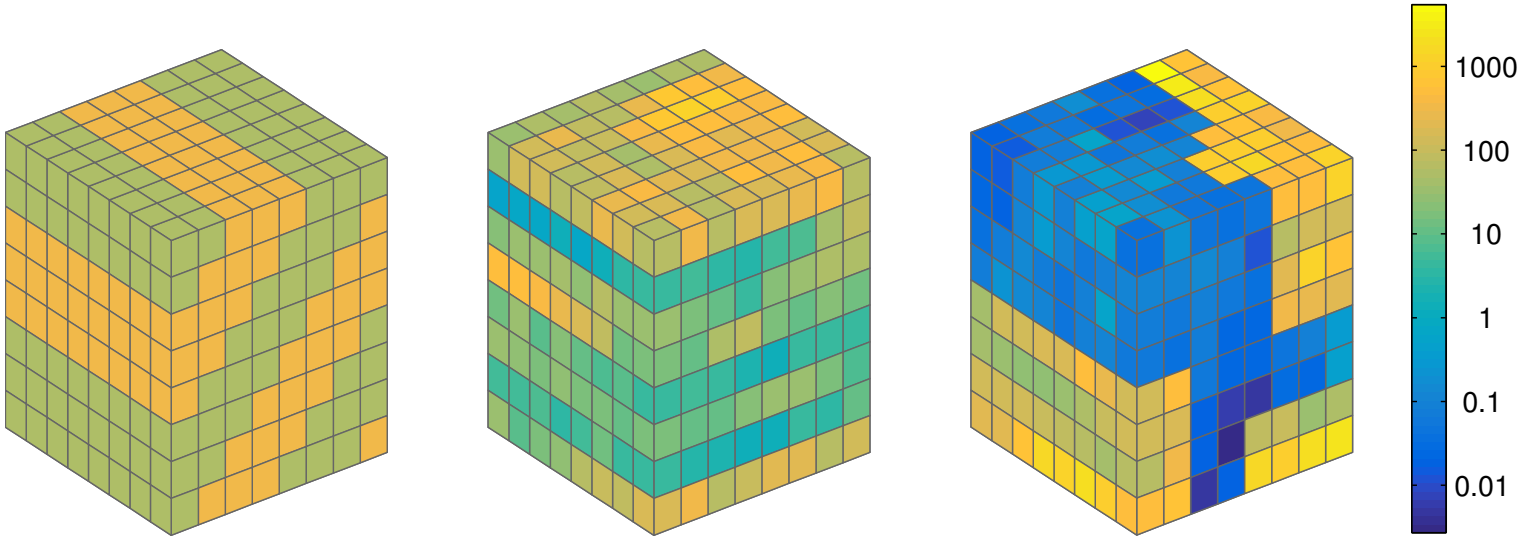
$$\mathbf{K}_s = \begin{bmatrix} 17.23 & 0 \\ 0 & 17.23 \end{bmatrix}$$

$$\mathbf{K}_p = \begin{bmatrix} 17.25 & -2.25 \\ -2.25 & 17.25 \end{bmatrix}$$

$$\mathbf{K}_s = \begin{bmatrix} 15.0 & 0 \\ 0 & 20.0 \end{bmatrix}$$

$$\mathbf{K}_p = \begin{bmatrix} 15.0 & 0 \\ 0 & 20.0 \end{bmatrix}$$

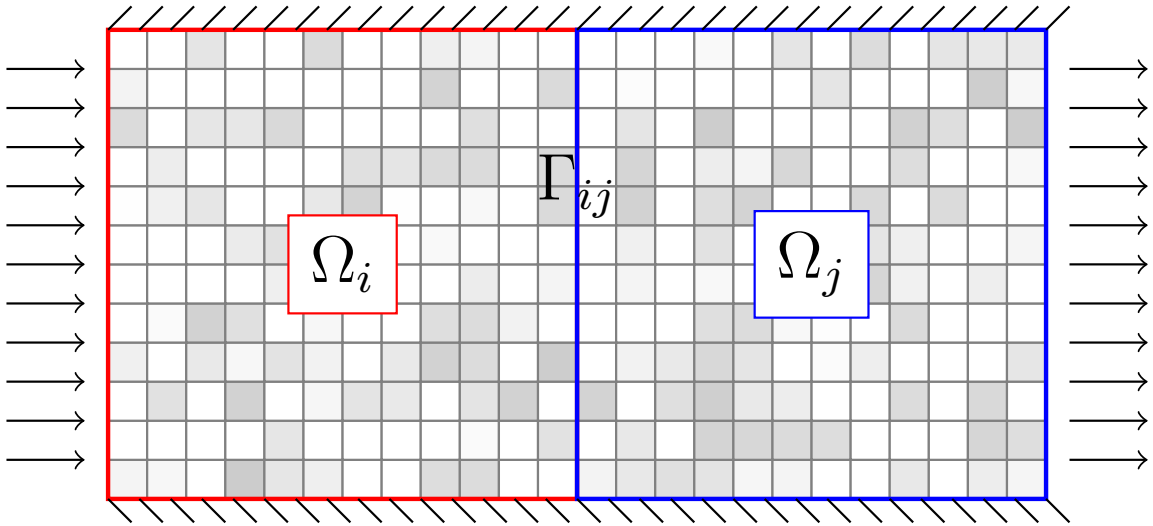
Example: ratio of fine and coarse-scale fluxes



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

Outline

Local transmissibility upscaling

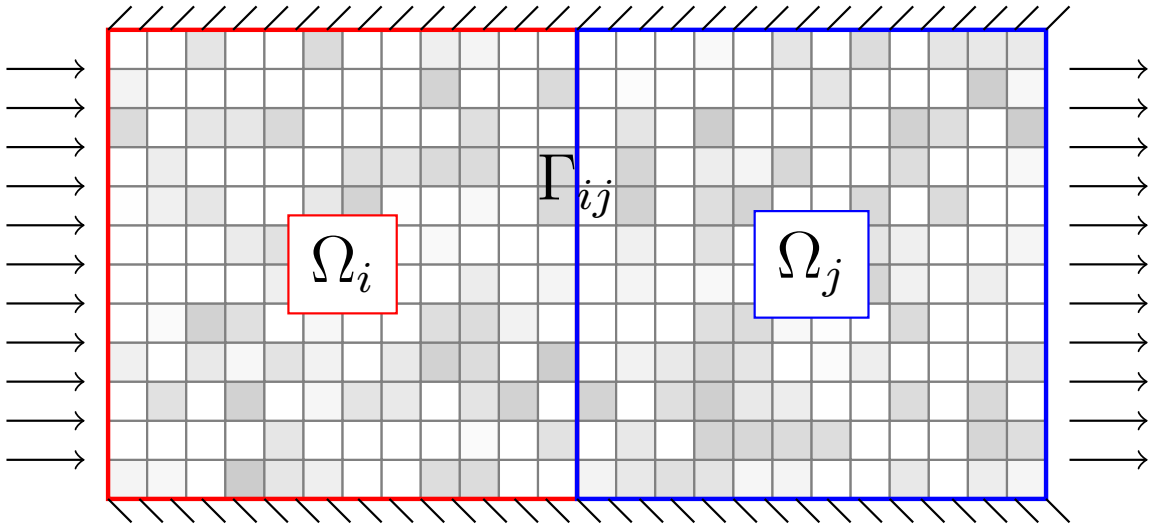


Instead of $\mathbf{K}^*$, we seek block transmissibilities T^*_{ij} satisfying

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

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

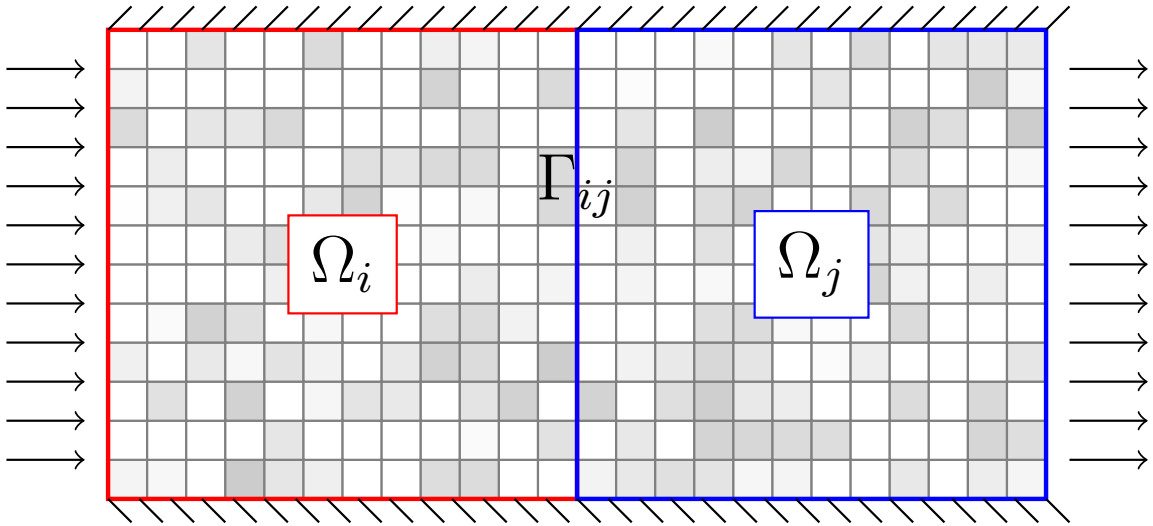
Local transmissibility upscaling



Construct coarse grid:

```
CG = generateCoarseGrid(G, q);
CG = coarsenGeometry(CG);
```

Local transmissibility upscaling



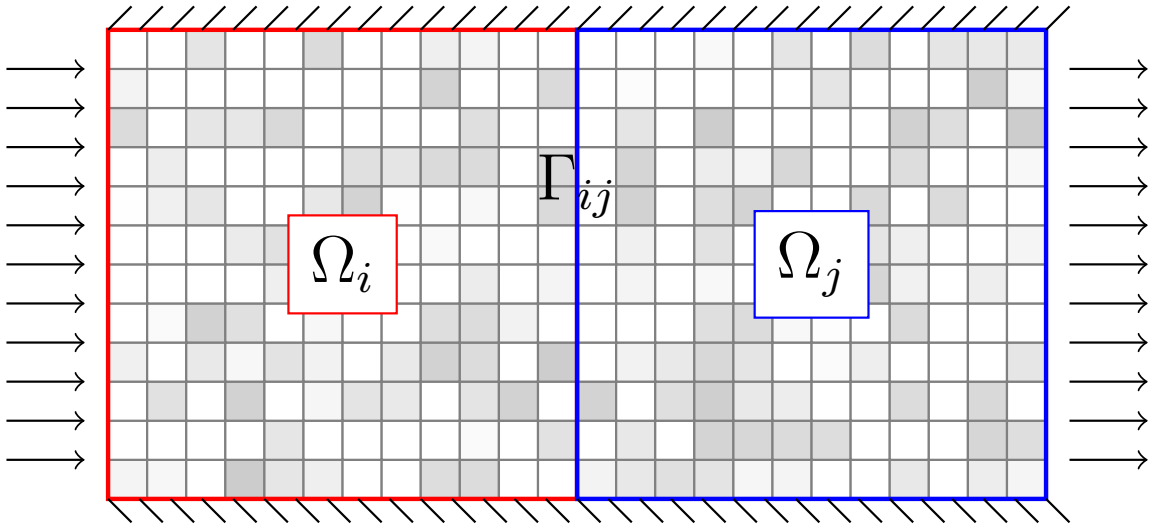
Construct coarse grid:

```
CG = generateCoarseGrid(G, q);
CG = coarsenGeometry(CG);
```

Find all faces in the fine grid that lie on Γ_{ij}:

```
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;
```

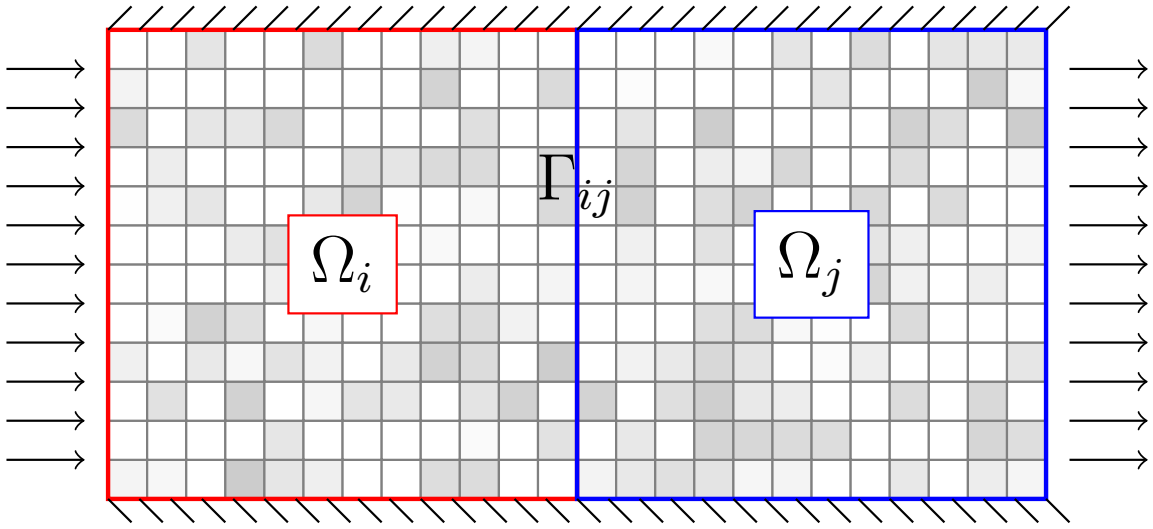
Local transmissibility upscaling



Using pressure drop and sealing boundary conditions:

```
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();
```

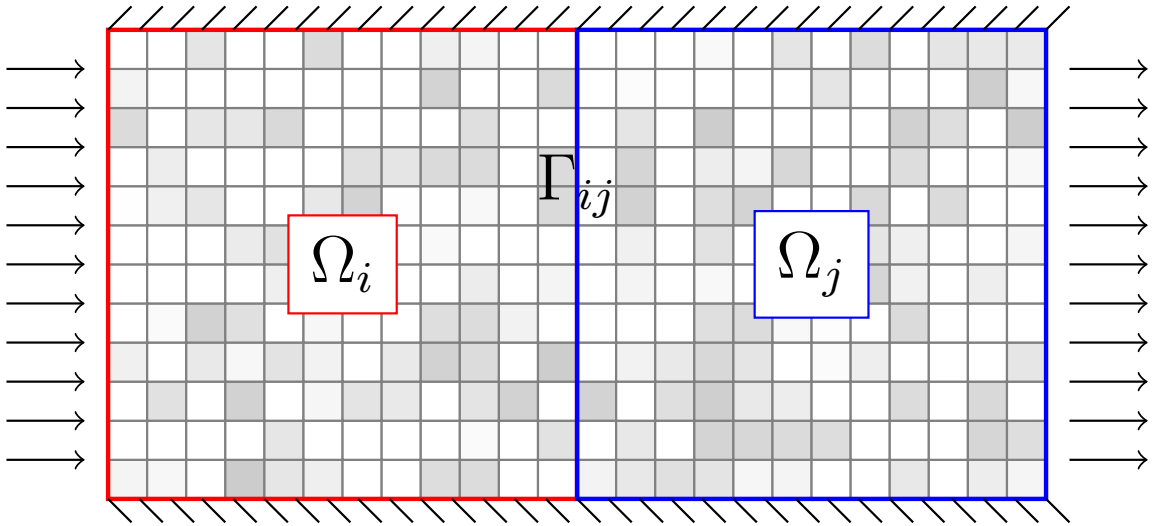
Local transmissibility upscaling



Compute average pressure values and invert Darcy's law:

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

Local transmissibility upscaling



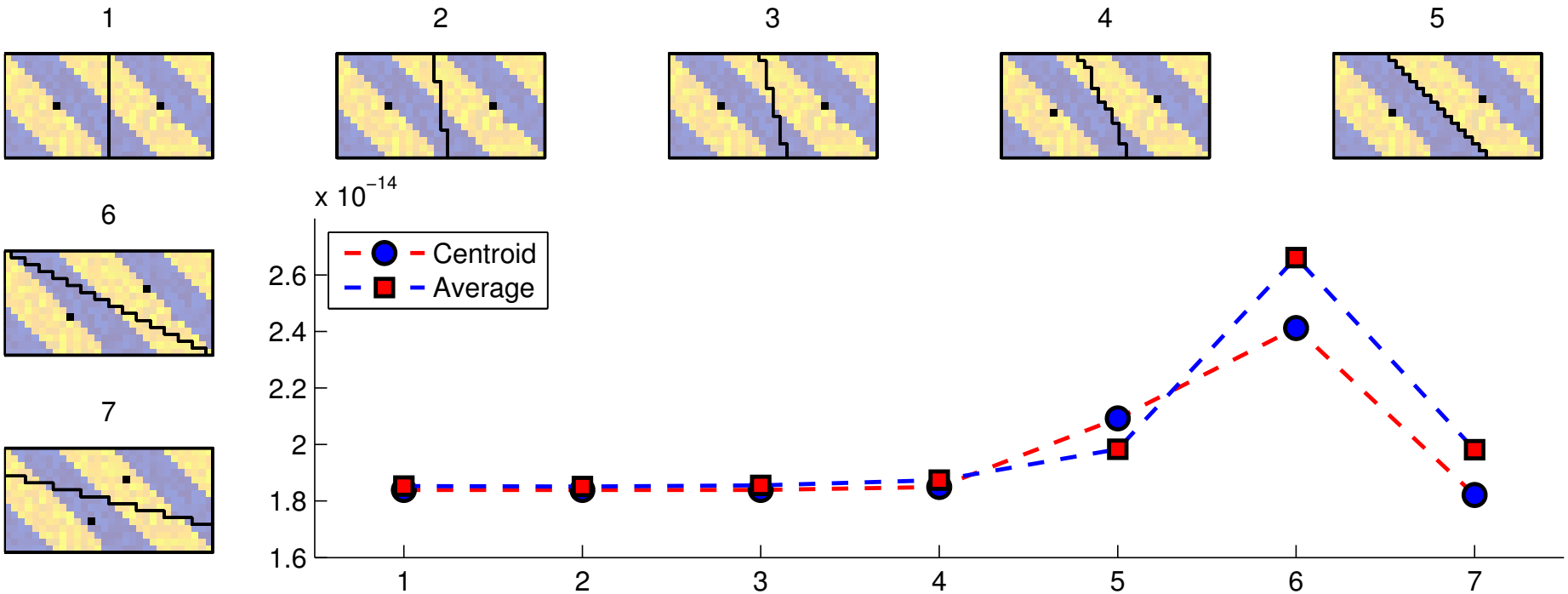
Compute average pressure values and invert Darcy's law:

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

Alternatively, use pressure at e.g., the block centroids:

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

Example: rotated two-block configuration



Positive transmissibilities

Transmissibilities should be positive to ensure that

$$\sum_j T_{ij}^*(p_i - p_j) = q_i,$$

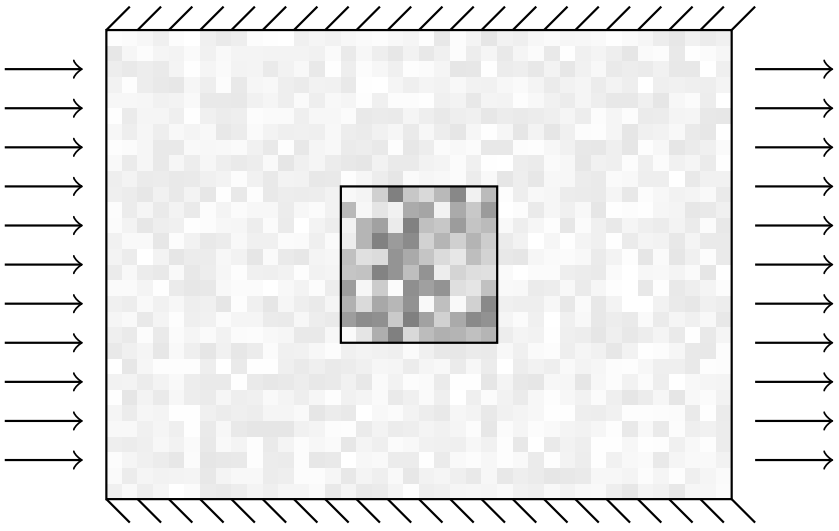
will reproduce the net grid block pressures $p_\ell = \frac{1}{|\Omega_\ell|}\int_{\Omega_\ell} p\, d\vec{x}$

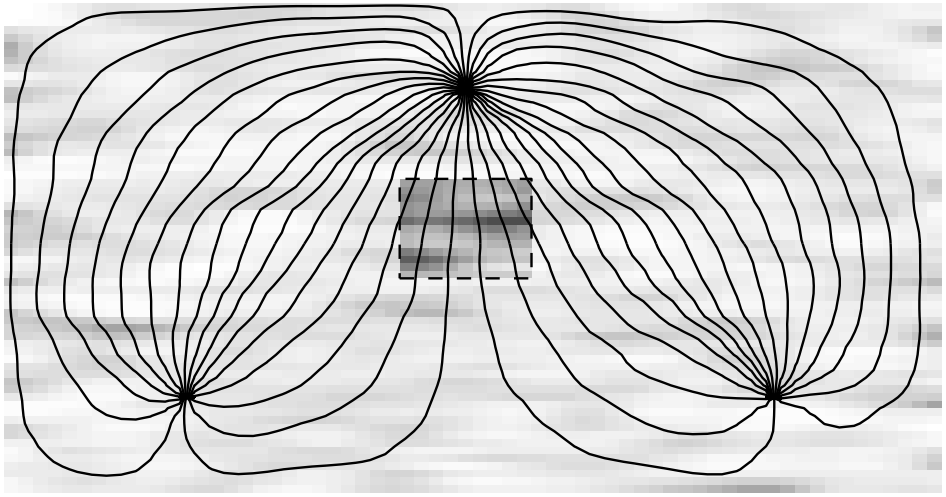
Not always the case:

- if the chord between the cell centroids does not cross the coarse interface
- if the petrophysical values are too heterogeneous (e.g., SPE 10 model)
- unique T^* may not exist since the upscaling problem is generally not well-posed

Trick of the trade: set $T_{ij} = \max(T_{ij}, 0)$

Oversampling, global, and local-global methods



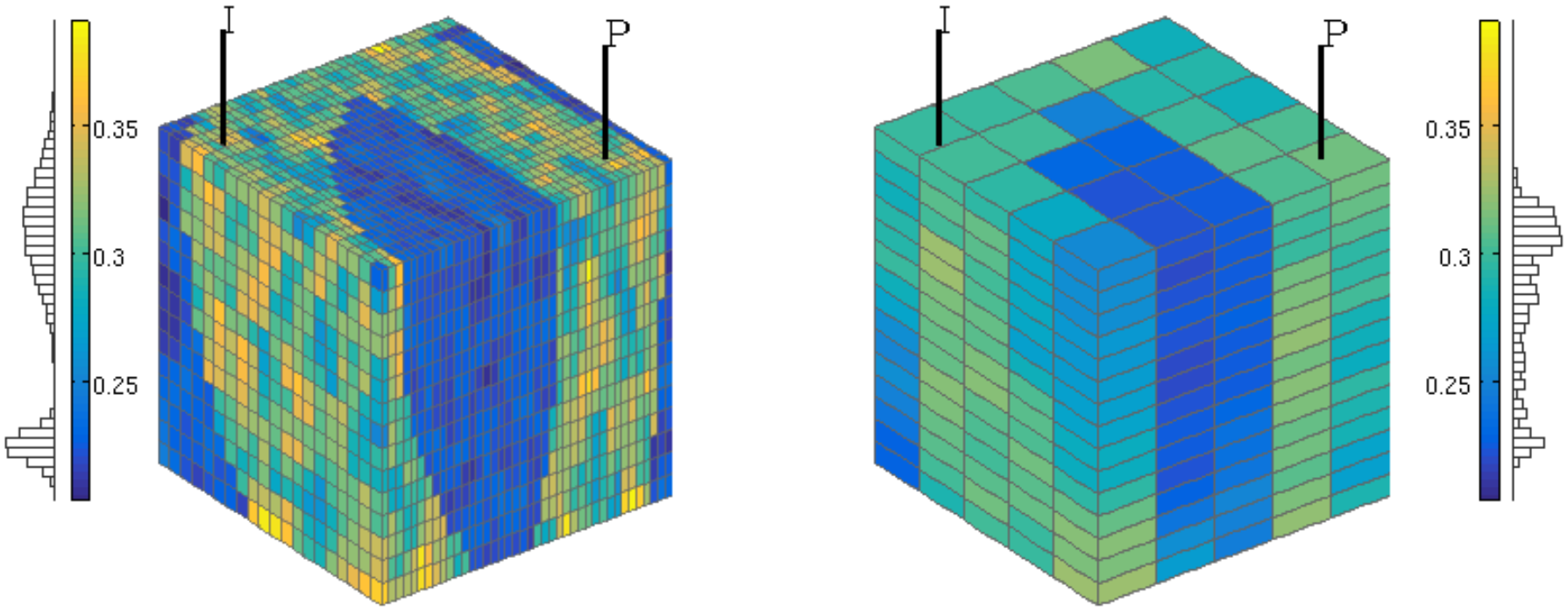


- Oversampling: lessen impact of the choice of b.c.
- Global: use global flow solution (*generic* or *specific*)
- Local-global: bootstrapping method, global flow problem solved on coarse grid
- Various fallback strategies to prevent/reduce negative values

In MRST: `upscaleTrans`

Outline

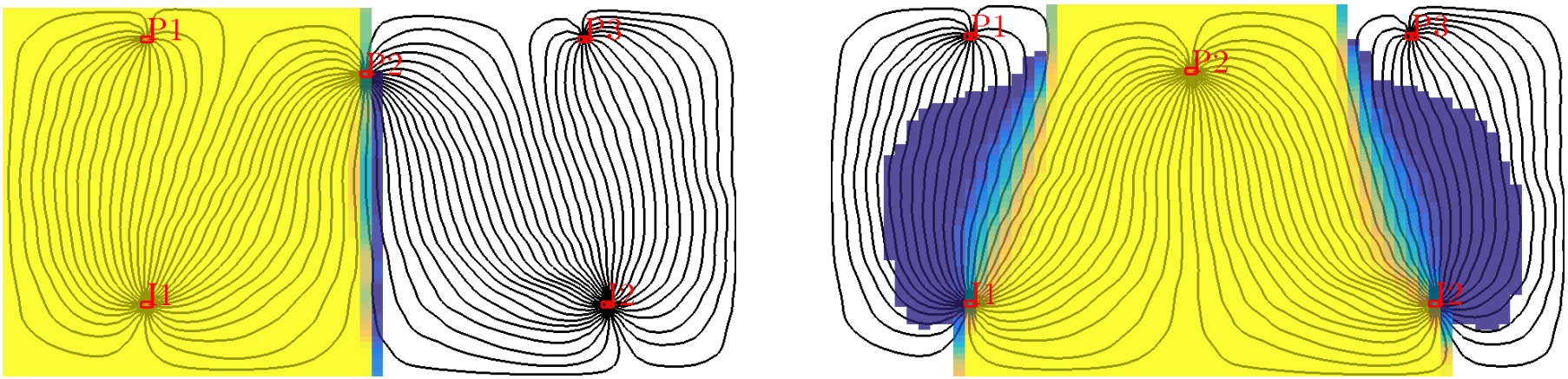
Example: model with two rock types



- Two facies with contrasting properties
- Vertical wells diagonally opposite in high-permeable part of model
- Two methods: local upscaling of $\mathbf{K}$, global generic upscaling of T

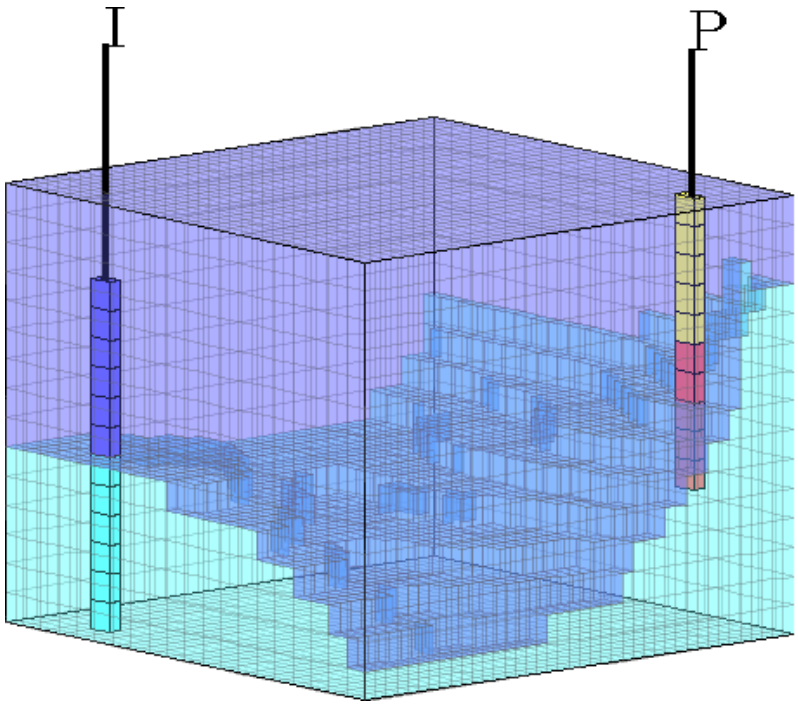
Example: model with two rock types

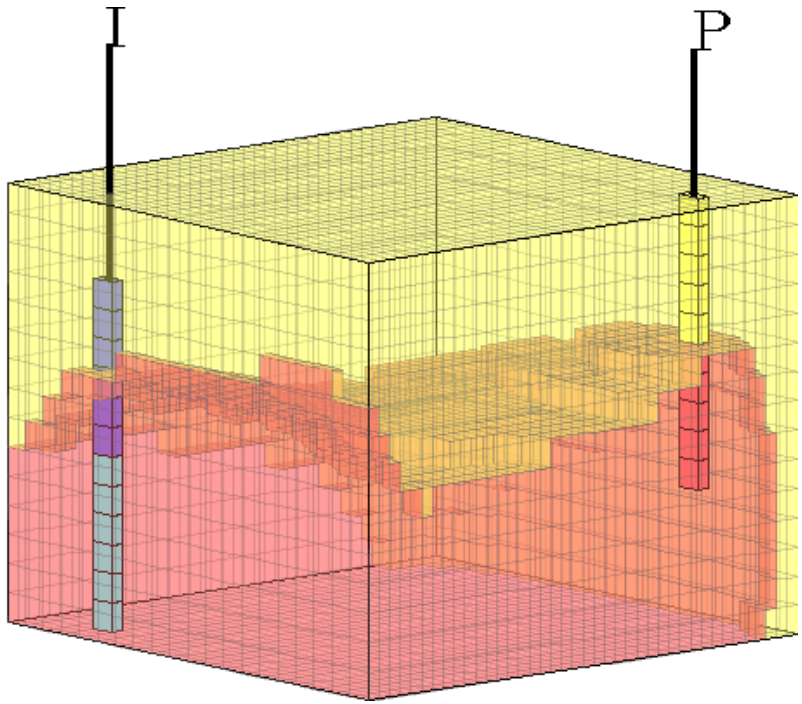
Quality assurance – flow diagnostics



Influence regions, from which one can e.g., compute well-allocation factors.

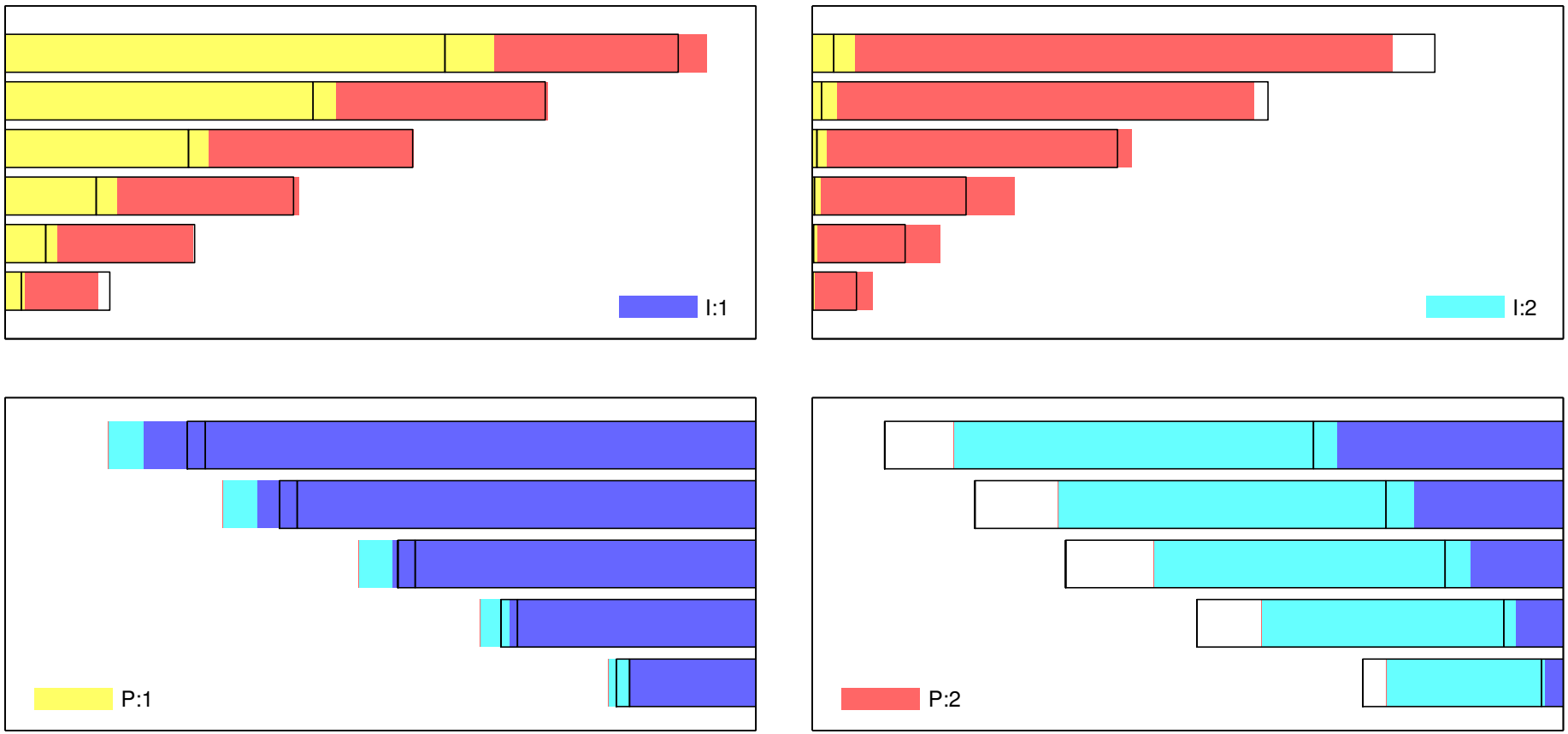
Example: model with two rock types





Example: model with two rock types

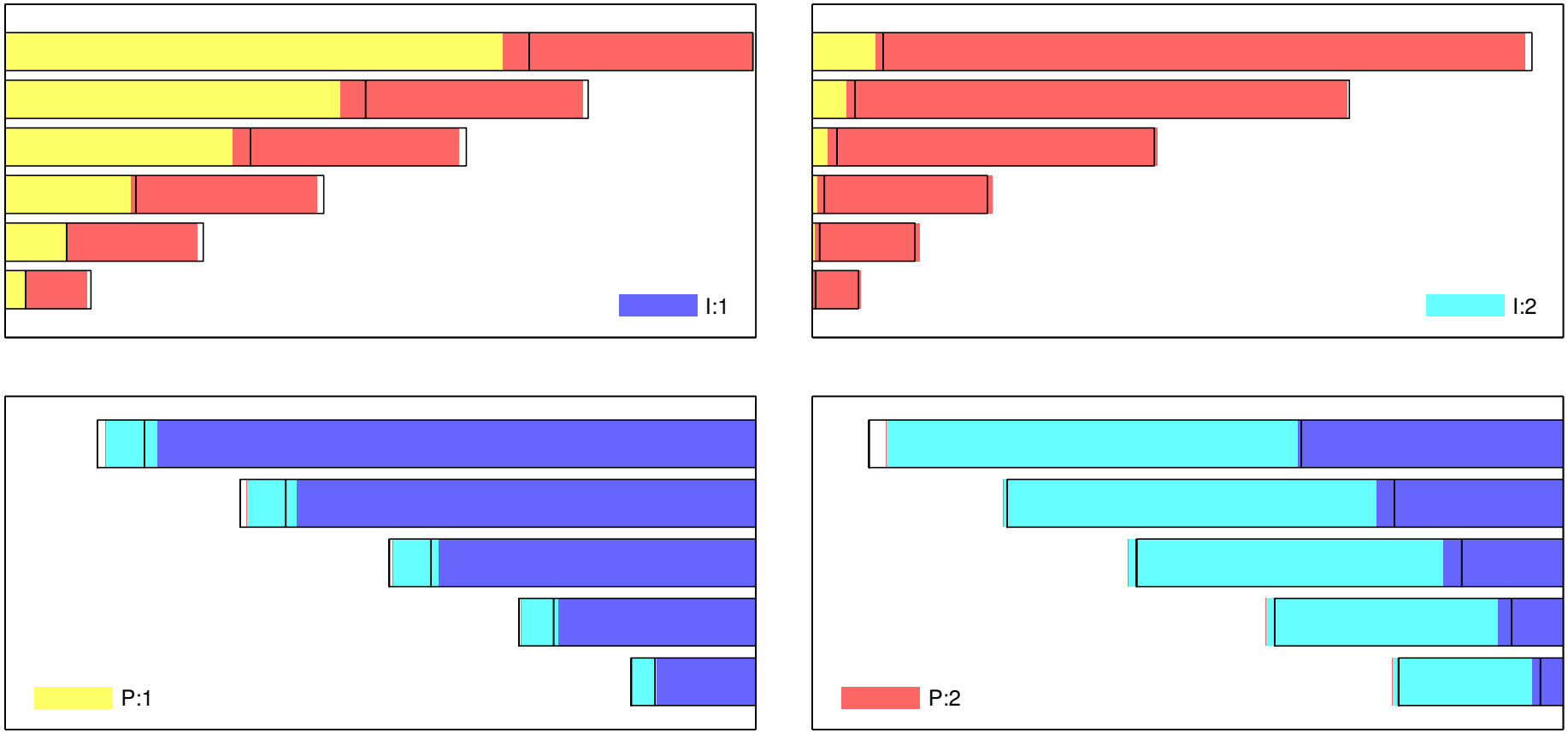
Local flow-based upscaling of $\mathbf{K}$:



Cumulative well-allocation factors computed on fine model (solid lines) and upscaled model (colorbars)

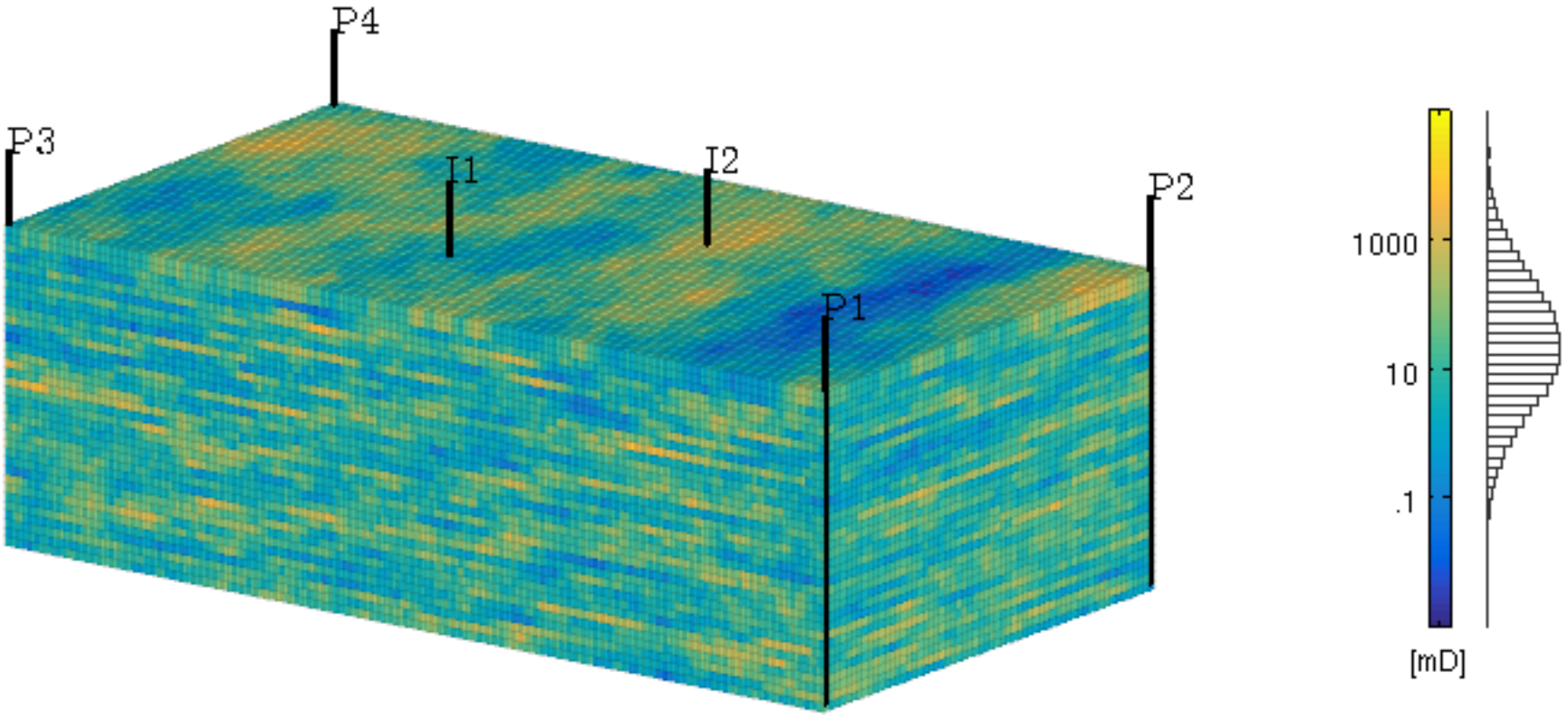
Example: model with two rock types

Global generic upscaling of T + upscaling of well indices:



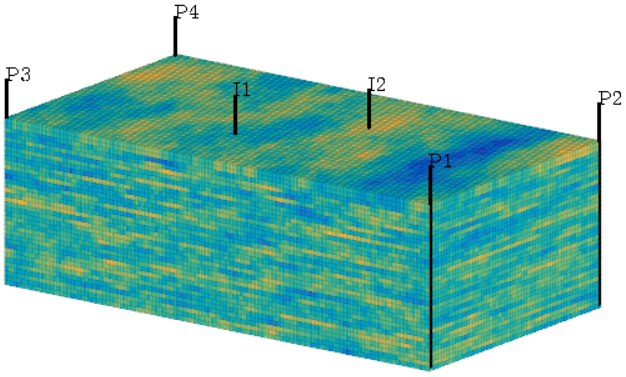
Cumulative well-allocation factors computed on fine model (solid lines) and upscaled model (colorbars)

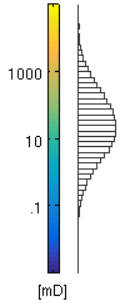
Example: SPE 10 benchmark

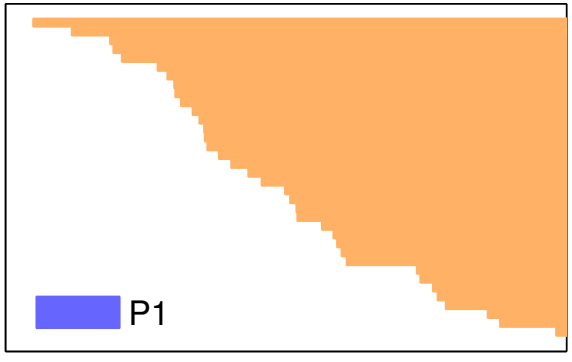


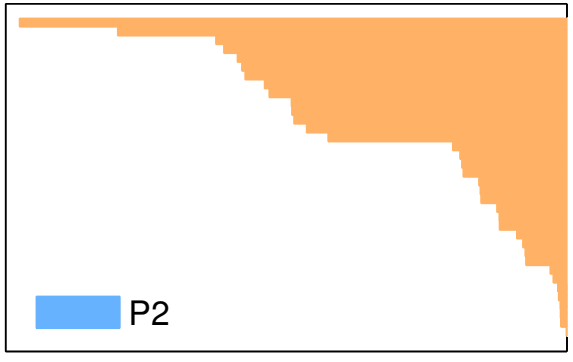
Tarbert formation (shallow marine, $220 \times 60 \times 36$) upscaled to a $22 \times 6 \times 12$ coarse model by four different methods. All wells controlled by bottom-hole pressure

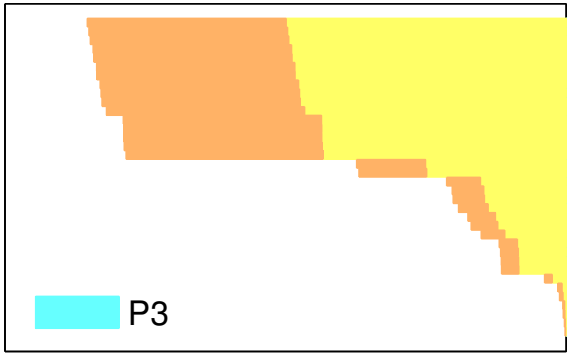
Example: SPE 10 benchmark

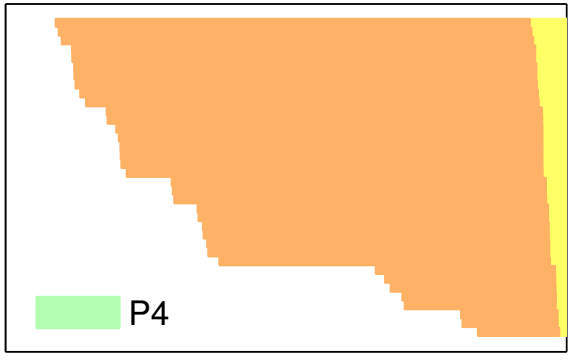


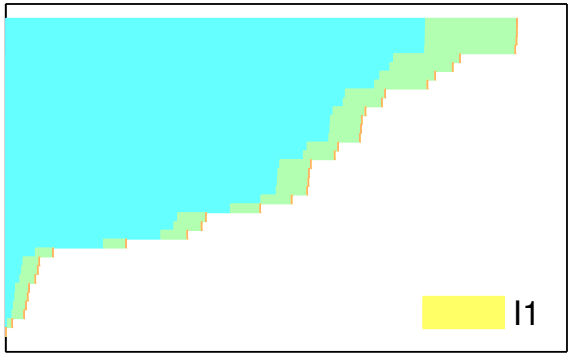


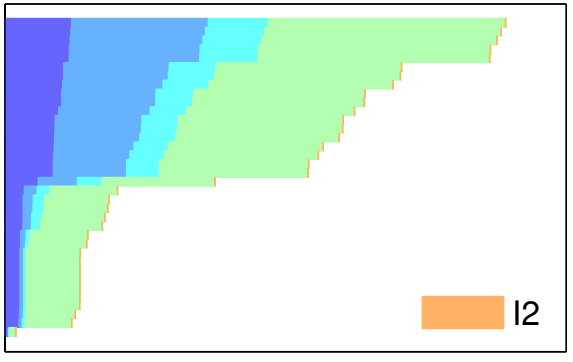




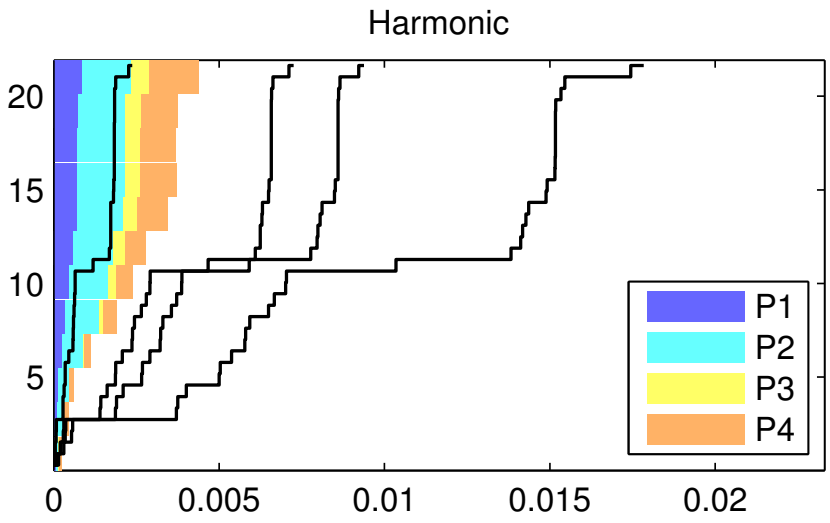


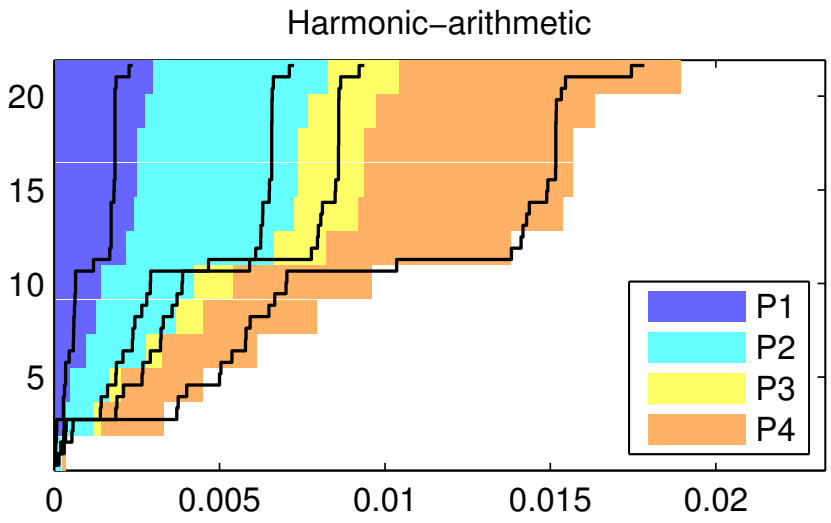


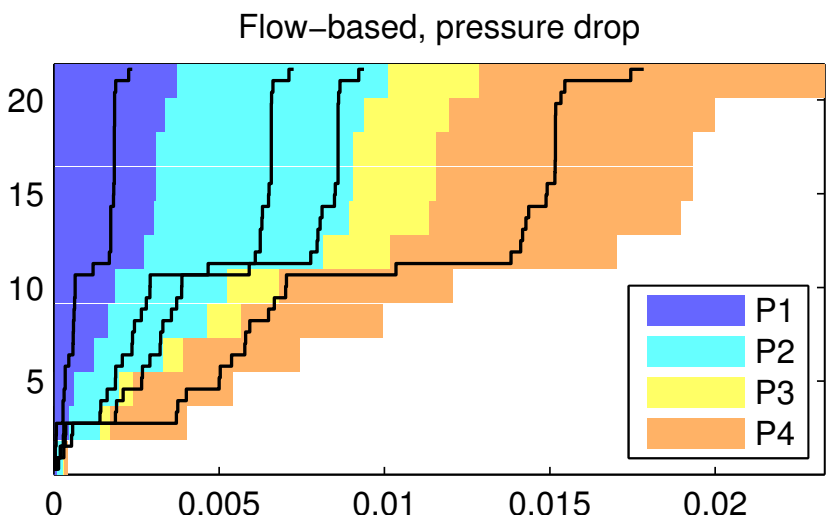


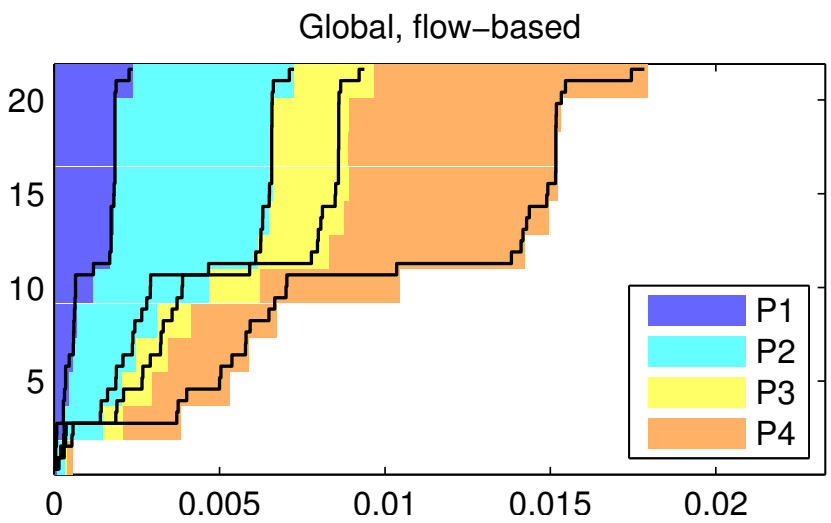


Example: SPE 10 benchmark

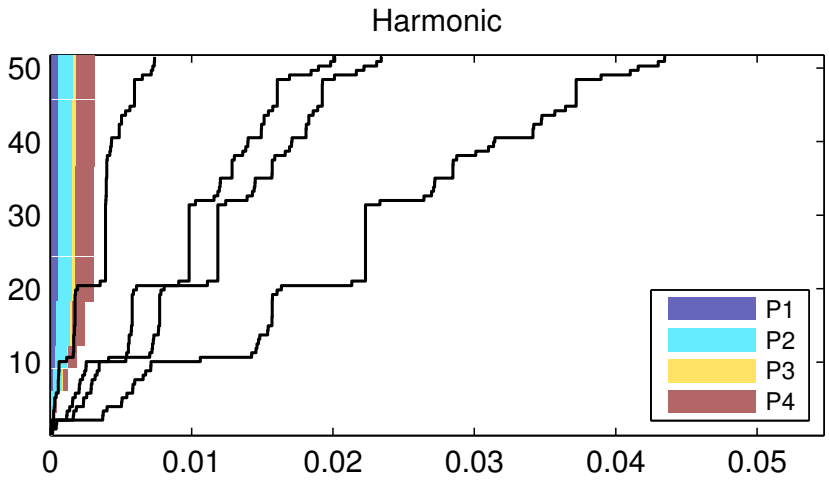


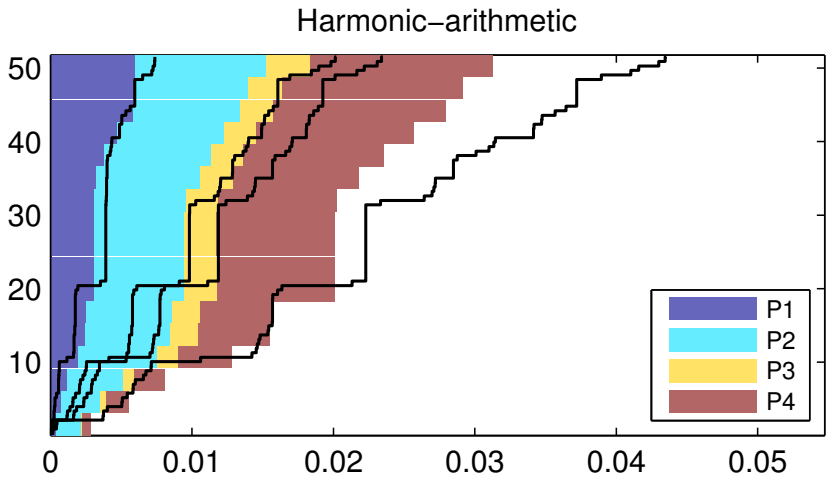


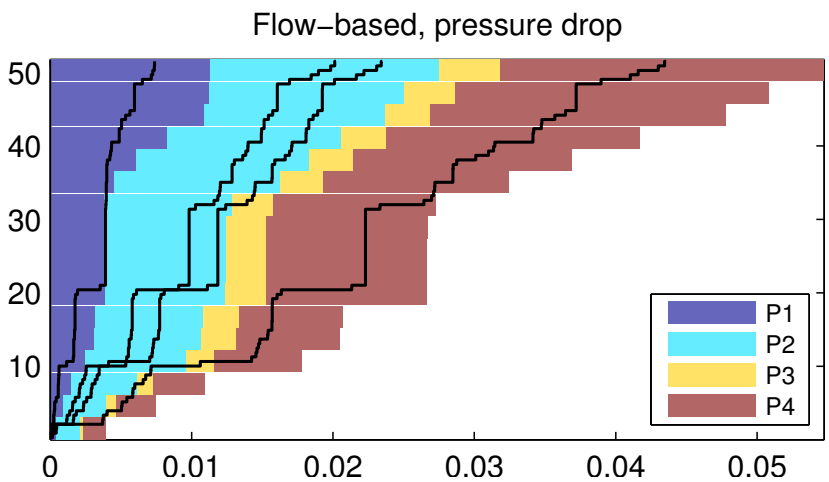


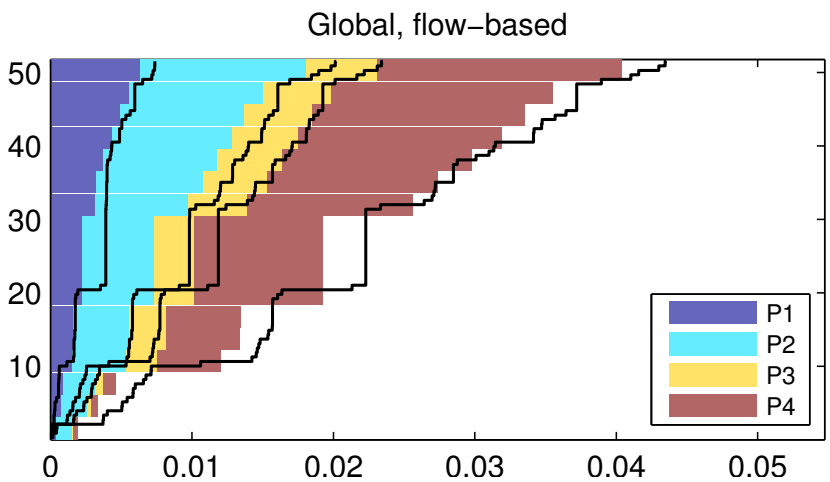


Example: SPE 10 benchmark









Summary

- Finding accurate and robust upscaling is difficult, no single method is unequivocally better than others
- Best choice depends upon purpose: aggressive coarsening with specific method if you only simulate a given scenario, otherwise pick a robust method
- Try different methods, starting by simple averaging to bound your values
- Try different coarse grids
- Check validity using single-phase flow physics (e.g., flow diagnostics) to exclude multiphase effects