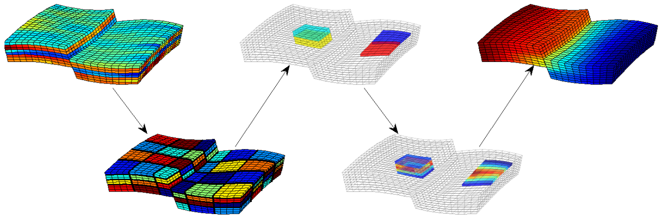
Flow-Based Coarsening for Multiscale Simulation of Transport in Porous Media

Knut–Andreas Lie, SINTEF, Norway



Workshop on Averaging, Upscaling, and New Theories
in Porous Media Flow and Transport
Bergen, October 14-15, 2010

What is multiscale simulation?

Generally:

Methods that incorporate fine scale information into a set of coarse scale equations in a way which is consistent with the local property of the differential operator

Herein:

Multiscale pressure solver (upscaling + downscaling in one step)

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

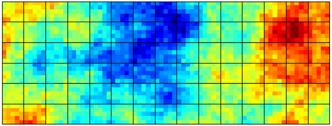
+ Transport solver (on fine, intermediate, or coarse grid)

$$\phi \frac{\partial S}{\partial t} + \nabla\Big(\vec{v} f(S)\Big) = q$$

= Multiscale simulation of models with higher detail

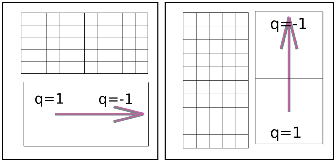
What is multiscale simulation?

Coarse partitioning:



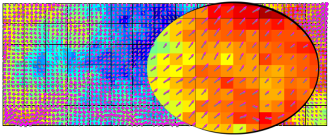
$\Downarrow$

Local flow problems:



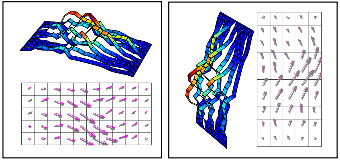
$\Rightarrow$

Flow field with subresolution:

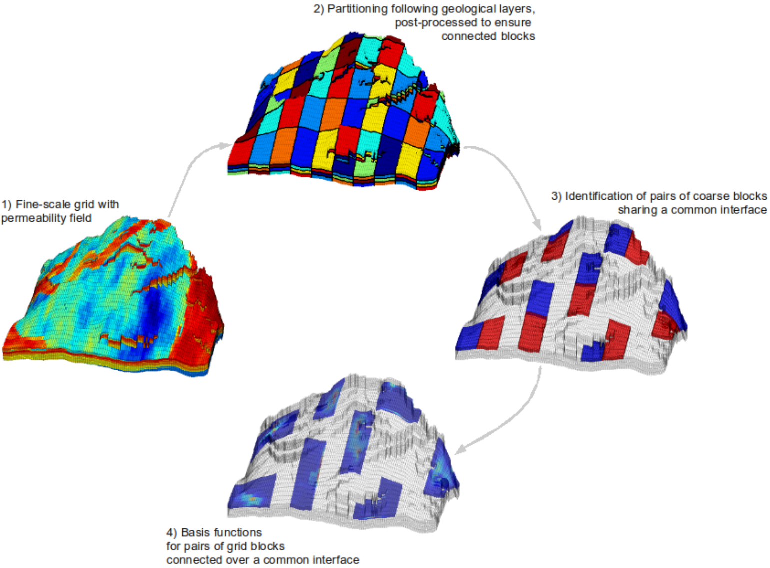


$\Uparrow$

Flow solutions $\longrightarrow$ basis functions:

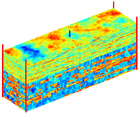


What is multiscale simulation?



What can you do with it?

Example 1: Model 2 of SPE 10

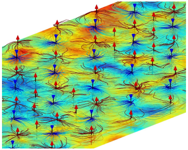


60 × 220 × 85 cells

Inhouse code from 2005 (TPFA):
- multiscale: 2 min and 20 sec
- multigrid: 8 min and 36 sec

Matlab/C solver (2010):
- ms-mimetic: 5–6 min

Example 2: History matching

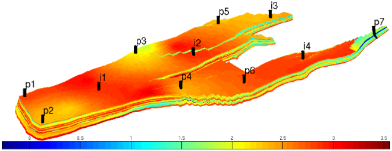


7 years: 32 injectors, 69 producers, 1 mill cells

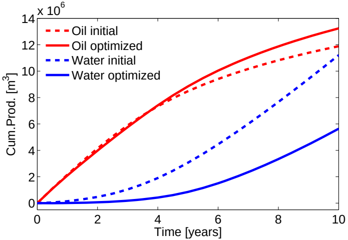
Generalized travel-time inversion + multiscale:
7 forward simulations, 6 inversions

Solver	CPU-time (wall clock) Total	Pres.	Transp.
Multigrid	39 min	30 min	5 min
Multiscale	17 min	7 min	6 min

Example 3: Rate optimization



Reservoir geometry from a Norwegian Sea field



Forward simulations:
- 44 927 cells, 20 time steps, < 5 sec in Matlab
- ∼ 100 times speedup

State-of-the-art

Capabilities:

- ✓ Incompressible (two-phase) flow
- ✓ Cartesian / unstructured grids
- ✓ Realistic flow physics $\Rightarrow$ iterations
  - Correction functions + smoothing
  - Residual formulation + domain decomposition
- ✓ Pointwise accuracy $\Rightarrow$ iterations

State-of-the-art / What is missing?

Capabilities:

- ✓ Incompressible (two-phase) flow
- ✓ Cartesian / unstructured grids
- ✓ Realistic flow physics $\Rightarrow$ iterations
  - Correction functions + smoothing
  - Residual formulation + domain decomposition
- ✓ Pointwise accuracy $\Rightarrow$ iterations

Not yet there:

- Compressible three-phase black-oil + non-Cartesian grids
- Fully implicit formulation
- Parallelization
- Compositional, thermal, ...
- Efficient and robust transport solvers

Transport solvers on coarse grids

Goal:
Given the ability to model velocity on geomodels and transport on coarse grids: Find a suitable coarse grid that best resolves fluid transport and minimizes loss of accuracy.

Formulated as the minimization of two measures:

1. the *projection error* between fine and coarse grid
2. the *evolution error* on the coarse grid

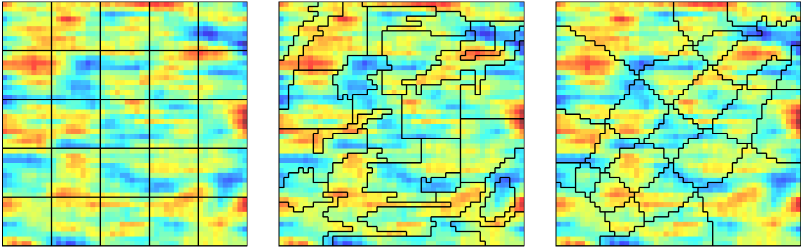
Difficult to formulate a practical and well-posed minimization problem for optimal coarsening $\longrightarrow$ ad hoc algorithms

Coarsening by amalgamation

Amalgamation of cells:

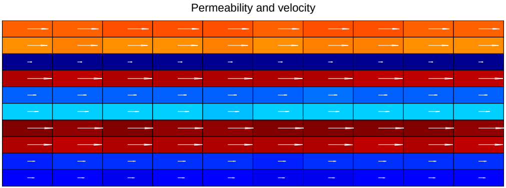
- coarse grid represented as partition vector: cell c_i in the fine grid is in coarse block B_j if $p_i = j$
- coarsening process steered by a set of admissible and feasible amalgamation directions

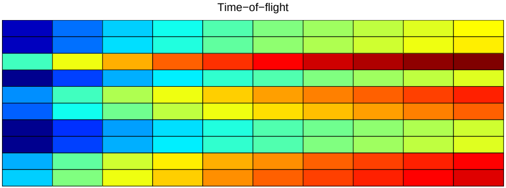
50×50 lognormal permeability:

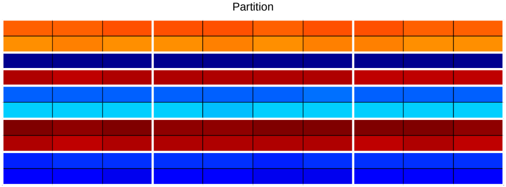


regular: 25 blocks — nonuniform: 26 blocks — isocontours [p]: 26 blocks

Motiviation: layered reservoir







Heuristic minimization algorithms

Formulated using a set of:

sources create a partition vector based on grid topology, geometry, flow-based indicator functions, error estimates, or expert knowledge supplied by the user, thereby introducing the feasible amalgamation directions

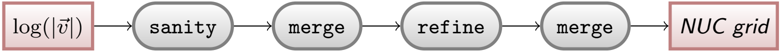
`filters` take a set of partition vectors as input and create a new partition as output, by

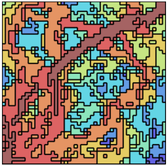
- combining/intersecting different partitions
- performing sanity checks, ensuring connected partitions etc
- modifying partition by merging small blocks or splitting large blocks

Non-uniform coarsening

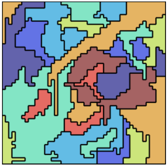
Aarnes, Efendiev & Hauge:

Use flow velocities to make a nonuniform grid in which each coarse block admits approximately the same total flow.

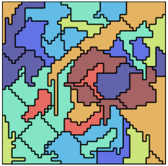




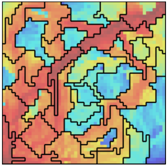
Partition: 304 blocks



Merging: 29 blocks

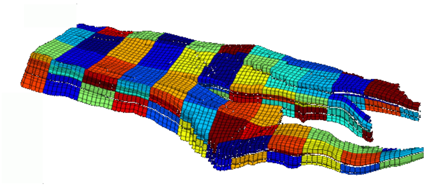


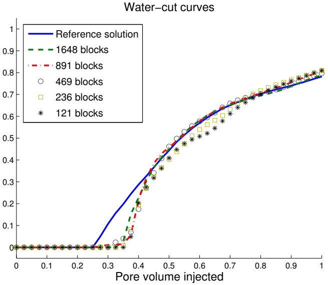
Refinement: 47 blocks

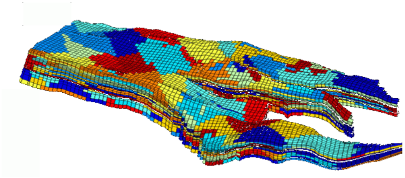


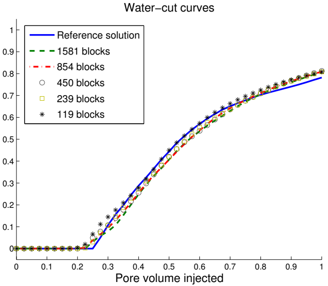
Merging: 39 blocks

Example: reservoir model from Norwegian Sea









Abstracting the NUC algorithm

Underlying principles:

- Minimize heterogeneity of flow field inside each block

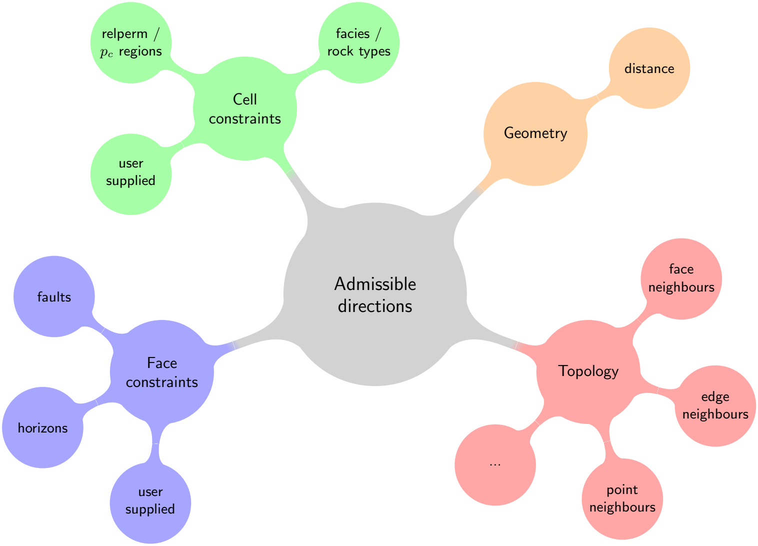
$$\min_{B_j}\Big(\sum_{p_i=j}|I_1(c_i)-I_1(B_j)|^p\,|c_i|\Big)^{\frac{1}{p}}, \qquad 1\le p\le\infty,$$

- Equilibrate indicator values over grid blocks

$$\min\Big(\sum_{j=1}^{N}|I_2(B_j)-\bar{I}_2(\Omega)|^p\,|B_j|\Big)^{\frac{1}{p}}, \qquad 1\le p\le\infty,$$

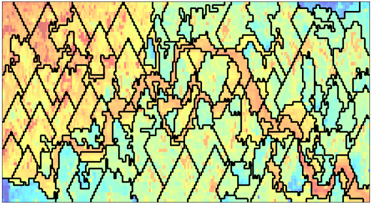
- Block size within prescribed lower and upper bounds

Amalgamation: admissible directions (neighbourship)

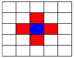


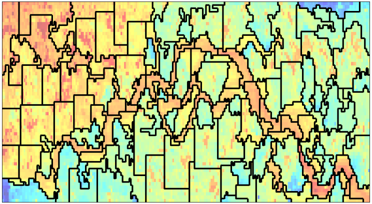
Example: extended neighbourship

Structured grid:

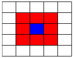


5-neighborhood

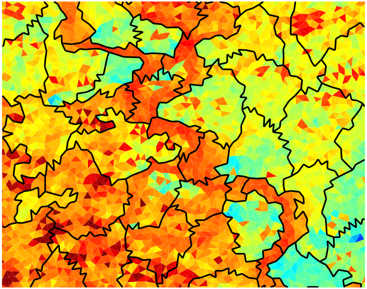


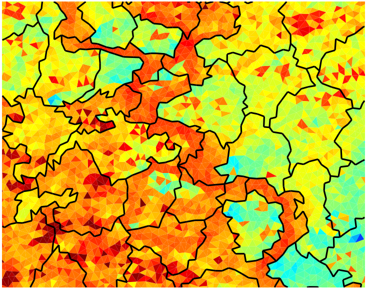


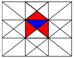
9-neighborhood

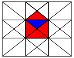


Triangular grid:

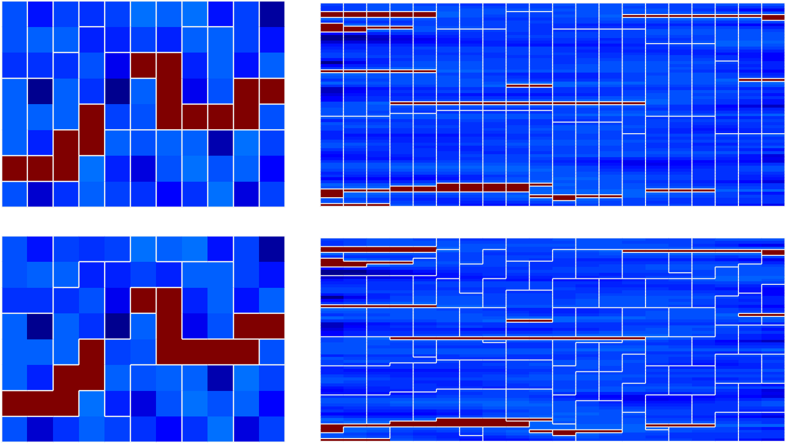




face neighbors 

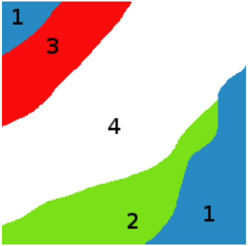
extended 

Example: restricted neighbourship (topology)

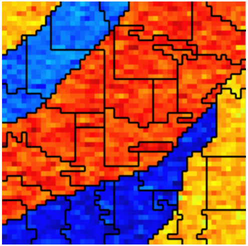


Upper row: $\mathcal{N}(c_{ij}) = \{c_{i,j-1}, c_{i,j+1}\}$
Lower row: $\mathcal{N}(c_{ij}) = \{c_{i,j\pm1}, c_{i\pm1,j}, c_{i\pm1,j\pm1}\}$

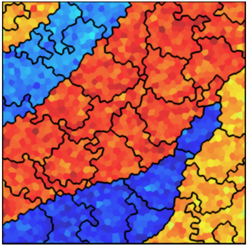
Example: restricted neighbourship (facies)



Facies distribution



Cartesian

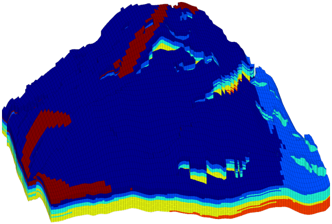


PEBI

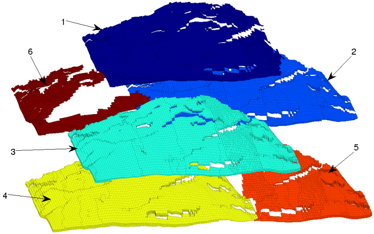
Constraining to facies / saturation regions:

- useful to preserve heterogeneity
- useful to avoid upscaling k_r and p_c curves

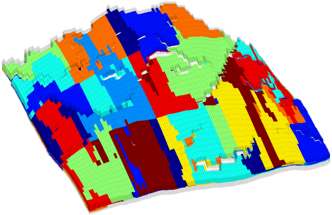
Example: restricted neighbourship (saturation regions)



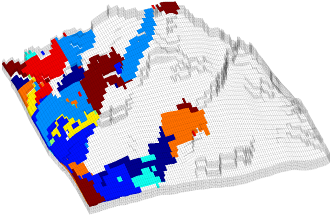
facies



facies separated



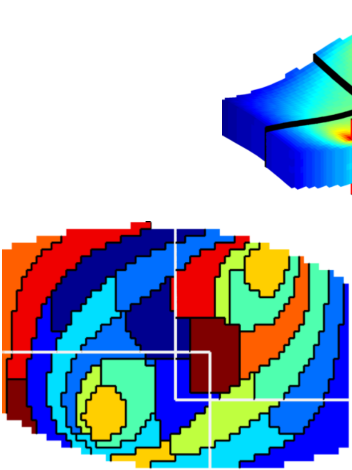
facies # 3



facies #6

Realization from SAIGUP study, coarsening within six different saturation regions

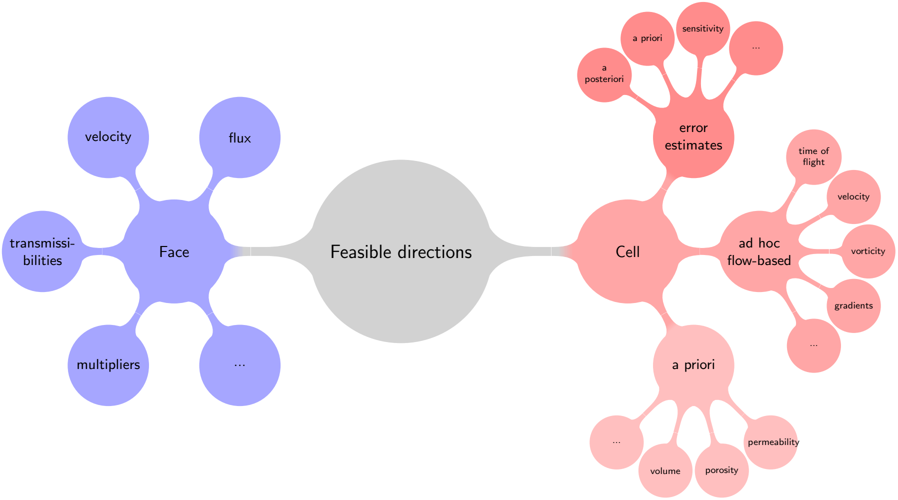
Example: restricted neighbourship (faults)



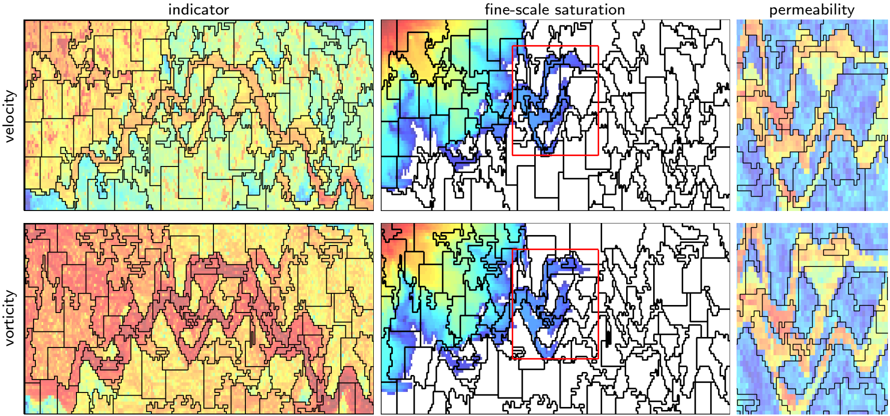
unconstrained: 26 blocks

constrained: 31 blocks

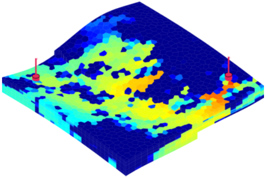
Amalgamation: feasible directions (indicators)



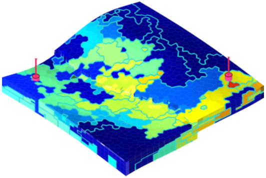
Example: flow-based indicators



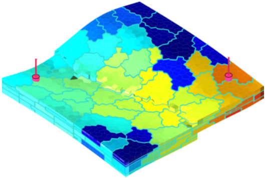
Example: flow-based indicators



reference solution
11 864 cells

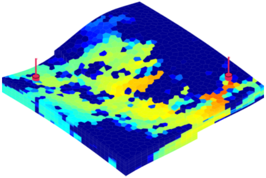


time-of-flight grid
127 blocks

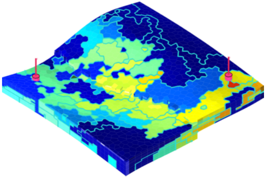


METIS grid
175 blocks

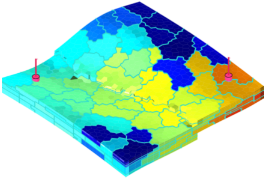
Example: flow-based indicators



reference solution
11 864 cells



time-of-flight grid
127 blocks



METIS grid
175 blocks

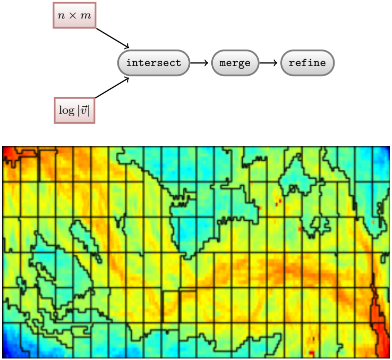
General observations:

- Time-of-flight is a better indicator than velocity
- Velocity is a better indicator than vorticity
- Vorticity is a better indicator than permeability
- ...

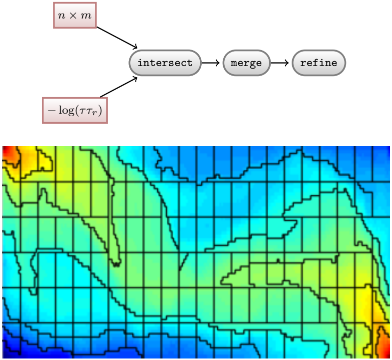
However, for smooth heterogeneities, the indicators tend to overestimate the importance of flow.

Example: hybrid methods

Velocity + Cartesian partition:

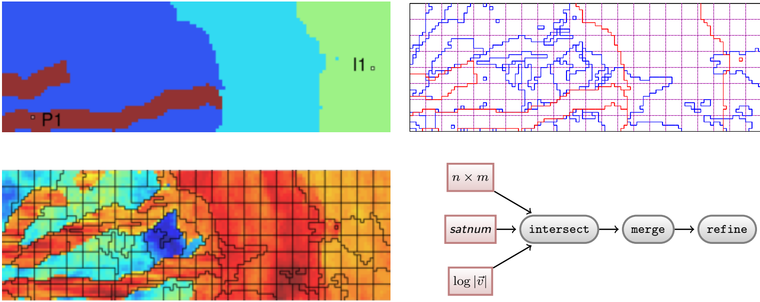


Time-of-flight + Cartesian partition:



Example: hybrid methods

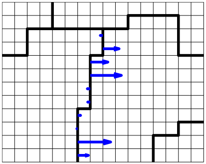
Satnum + velocity + Cartesian:



Coarse-grid discretisation

Bi-directional fluxes (upwind on fine scale):

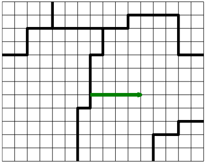
$$S_\ell^{n+1} = S_\ell^n - \frac{\Delta t}{\phi_\ell |B_\ell|}\Big[f(S_\ell^{n+1}) \sum_{\partial B_\ell} \max(v_{ij},0) - \sum_{k\neq\ell}\Big(f(S_k^{n+1}) \sum_{\Gamma_{k\ell}} \min(v_{ij},0)\Big)\Big].$$



This gives a centred scheme on the coarse scale

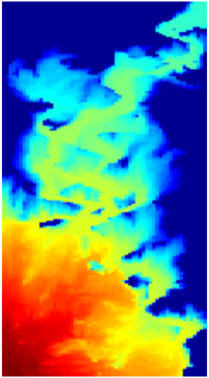
Net fluxes:

$$S_\ell^{n+1} = S_\ell^n - \frac{\Delta t}{\phi_\ell |B_\ell|} \sum_{k\neq\ell} \max\Big(f(S_\ell^{n+1}) \sum_{\Gamma_{k\ell}} v_{ij}, -f(S_k^{n+1}) \sum_{\Gamma_{k\ell}} v_{ij}\Big).$$

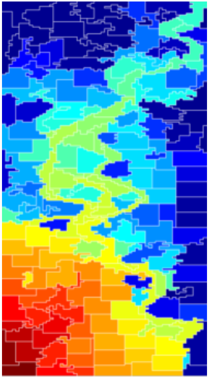


This gives an upwind scheme on the coarse scale

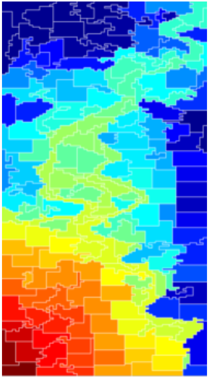
Coarse-grid discretisation: numerical diffusion



reference



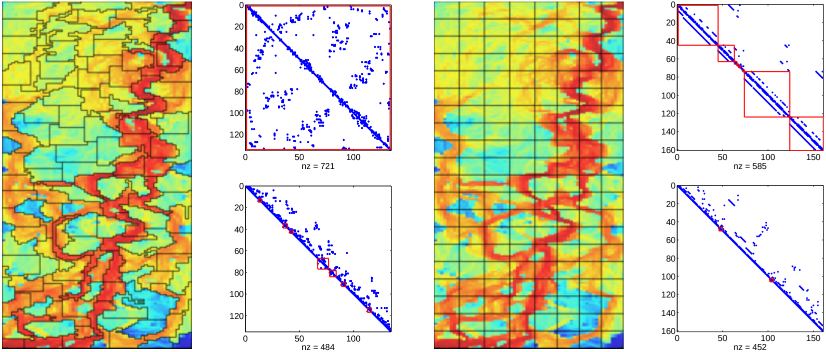
net fluxes



bi-directional fluxes

Layer 37 from SPE10

Coarse-grid discretisation: matrix structure



Layer 68 from SPE10

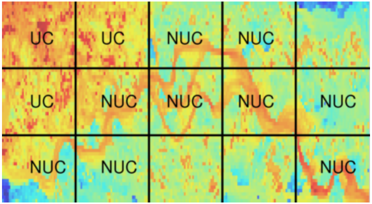
Coarse-grid discretisation: numerical errors

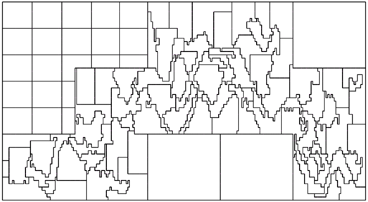
	Tarbert formation			Upper Ness formation		
	NUC/Cart	NUC	Cartesian	NUC/Cart	NUC	Cartesian
$E_s(\mathcal{PR}S_f, S_f)$	0.0941	0.1042	0.0911	0.1371	0.1355	0.1772
$E_s(\mathcal{P}S_c, S_f)$	0.1910	0.2426	0.1687	0.2124	0.2243	0.2305
$E_s(S_c, \mathcal{R}S_f)$	0.1599	0.2100	0.1381	0.1522	0.1683	0.1604
$E_w(w_c, w_f)$	0.0695	0.0773	0.0701	0.0609	0.0668	0.0982
$E_s(\mathcal{P}S_c, S_f)$	0.1607	0.1875	0.1619	0.1795	0.1862	0.2191
$E_s(S_c, \mathcal{R}S_f)$	0.1237	0.1459	0.1302	0.1135	0.1225	0.1486
$E_w(w_c, w_f)$	0.0473	0.0444	0.0647	0.0237	0.0325	0.0844
# blocks	217–261	233–312	264	205–241	220–303	264
mean	236	275	264	222	264	264
# faces: mean	1069	1363	1090	1070	1309	1090

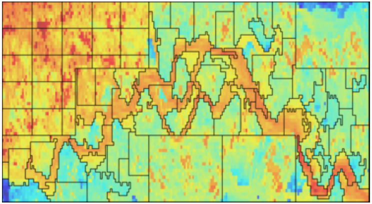
bi-directional fluxes (rows 2–4, blue) · net fluxes (rows 5–7, pink)

Average errors over all layers of the two formations in SPE10

Supervised coarsening



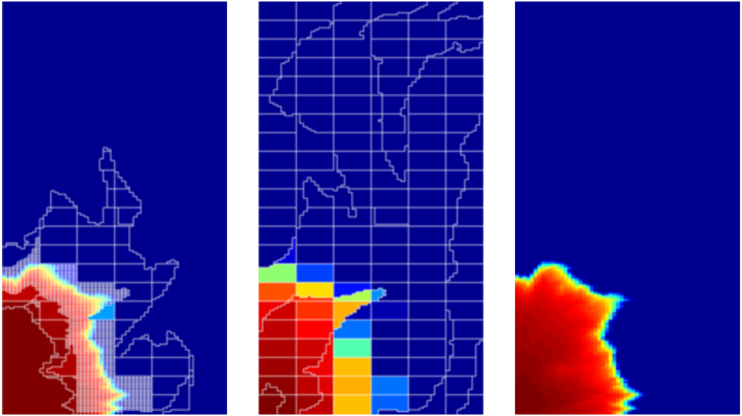




Layer 37 from SPE10

Dynamical adaption

After injection of 0.1 PVI

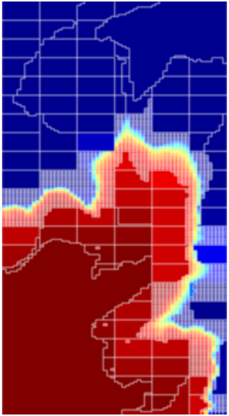


adaptive | coarse | fine grid

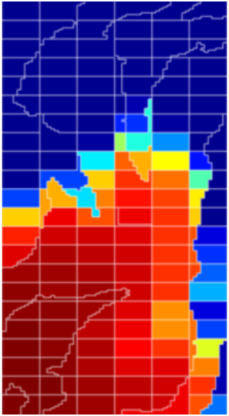
Layer 22 from SPE10

Dynamical adaption

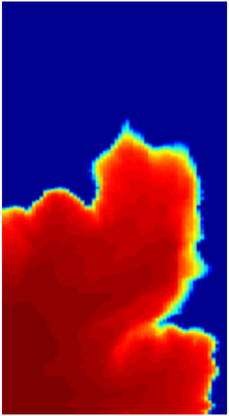
After injection of 0.5 PVI



adaptive



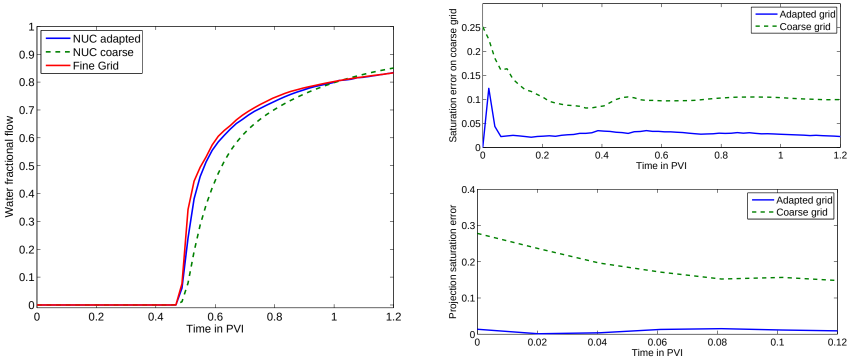
coarse



fine grid

Layer 22 from SPE10

Dynamical adaption



Layer 37 from SPE10

Questions

How can these methods be useful? For what purpose would you apply them?

- As robust upscaling methods?
- As alternative to upscaling and fine-scale solution?
- To provide flow simulation earlier in the modelling loop?
- To get 90% of the answer in 10% of the time?
- Fit-for-purpose solvers in workflows for ranking, history matching, planning, optimization, ...

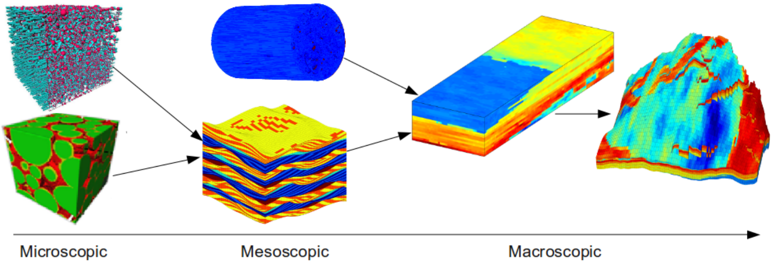
Questions

Which capabilities should we try to develop?

- More complex flow physics?
- Modelling of fault/fractures?
- Multiphysics formulations?
- Automated methods with goal-oriented error control?
- ...

What capabilities are sufficient for the methods to be more generally adopted?

Questions



Ulitmate vision: 'truly' multiscale methods, bridging 'all' scales?

Is incorporation of pore/core/facies models realistic?