The MATLAB Reservoir Simulation Toolbox

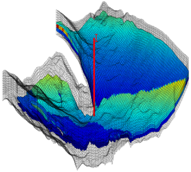
Bård Skaflestad

SIAM Geosciences, Long Beach, March 21–24, 2011

The Matlab Reservoir Simulation Toolbox (MRST)

The toolbox has the following functionality for rapid prototyping of solvers for flow and transport:

- grid structure, grid factory routines, input/processing of industry-standard formats, real-life and synthetic example grids
- petrophysical parameters and incompressible fluid models, conversion routines to/from SI and common field units, very simplified geostatistical routines
- routines for setting and manipulating boundary conditions, sources/sinks, and well models
- reservoir state (pressure, fluxes, saturations, compositions, . . .)
- visualisation routines for cell and face data (scalars)



Download

http://www.sintef.no/MRST/

Version 2011a was released on the 22nd of February, 2011, and can be downloaded under the terms of the GNU General Public License (GPL)

Software goals

- General framework for flow and transport in porous media
- Special focus on unstructured grids and multiscale methods

Research:

- Development of new solvers and discretization schemes
- Application to new scientific/engineering problems
- Preserve know-how and promote reuse of results from previous research
- Contribute to accelerating production of new research (by our peers)
- Benchmarking – compare different methods on standard test problems

Students:

- Develop students' intuition of porous media flow
- Make it easy to test, compare, and extend existing methods
- Textbook (with worked examples) in preparation

Why open-source and why in Matlab?

First of all, prototyping in a scripting language is much more effective than in traditional compiled languages (C/C++/FORTRAN)

- Explore alternative algorithms/implementations close to mathematics
- Gradually replace individual (or bottleneck) operations with accelerated editions callable from MATLAB
- Direct access to MATLAB environment and prototype whilst developing replacement components
- More experienced in Matlab/Octave than in e.g., Python,

Why free and open-source software:

- Research funded by Norwegian Research Council should be freely available
- Combined with publications: a means to collaborate and disseminate results, while protecting IP rights
- Support *reproducible research* and promote replicability

How is MRST designed?

The fundamental object in MRST is the grid:

- Data structure for geometry and topology
- Several grid factory routines
- Input of industry-standard (proprietary) format(s)

Physical quantities defined as dynamic objects in MATLAB

- Properties of medium (ϕ, **K**, net-to-gross, . . .)
- Reservoir fluids (ρ, μ, k_r, PVT, . . .)
- Driving forces (wells, boundary conditions, sources)
- Reservoir state (pressure, fluxes, saturations, etc)

All MRST operations accept, manipulate and produce objects of these types.

Physical quantities are assumed to be in SI units.

How is MRST designed?

MRST core

- routines for creating and manipulating grids and physical properties
- basic flow and transport solvers (sequential splitting) for incompressible and immiscible flow

Functionality is stable and not expected to change in future releases

Modules

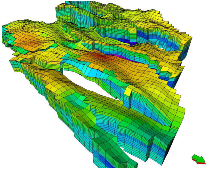
Similar to MATLAB's toolboxes. Implements more advanced solvers and tools:

- adjoint methods, experimental multiscale, fractures, MPFA, upscaling
- black-oil models, three-phase flow, vertically integrated models, ...
- streamlines, (flow-based) coarsening, ...
- Octave support, C-acceleration, ...

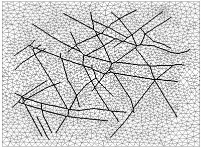
Some are stable. Some are constantly changing to support ongoing research.
New modules initiated by others are much welcome

Grids in MRST
Complex reservoir geometries

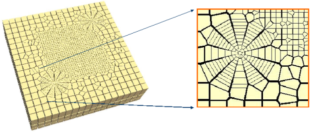
Corner point:



Tetrahedral:



PEBI:

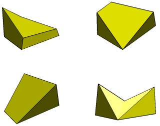


- Accurate modelling of features such as faults, fractures, or erosion requires grids that are flexible with respect to geometry.
- Industry-standard grids are often nonconforming and contain complex grid-cell connectivities, as well as skewed and degenerate cells
- There is a trend towards unstructured grids with general polyhedral cells
- Standard discretization methods produce wrong results on skewed and rough cells condition numbers
- *The representation of (un)structured grids affects the efficiency of numerical methods*

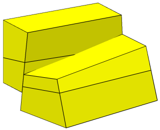
Grids in MRST

Cell geometries are challenging from a discretization point-of-view

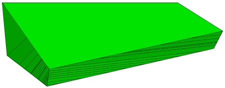
Skewed and deformed blocks:



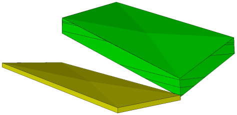
Non-matching cells:



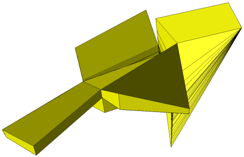
Many faces:



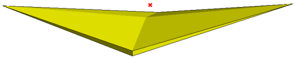
Small interfaces:



Difficult geometries:

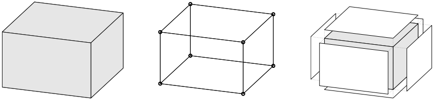


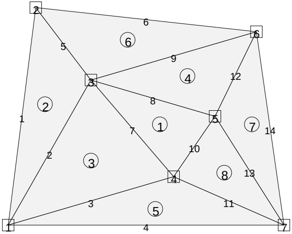
(Very) high aspect ratios:



$800 \times 800 \times 0.25$ m

Basic representation:



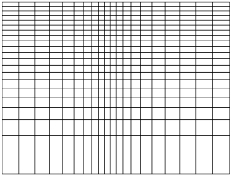


```
cells.faces =      faces.nodes =      faces.neighbors =
    1   10             1    1             0    2
    1    8             1    2             2    3
    1    7             2    1             3    5
    2    1             2    3             5    0
    2    2             3    1             6    2
    2    5             3    4             0    6
    3    3             4    1             1    3
    3    7             4    7             4    1
    3    2             5    2             6    4
    4    8             5    3             1    8
    4   12             6    2             8    5
    4    9             6    6             4    7
    5    3             7    3             7    8
    5    4             7    4             0    7
    5   11             8    3
    6    9             8    5
    6    6             9    3
    6    5             9    6
    7   13            10    4
    7   14            10    5
    :    :             :    :
```

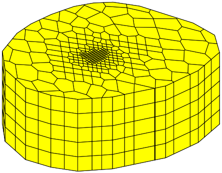
Choices in grid representation guided by utility and convenience in low-order finite-volume methods. Available geometric information typically limited to centroids, normals, areas, and volumes

Grids in MRST

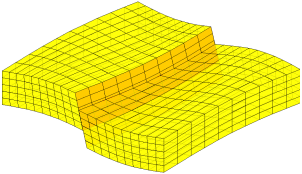
Examples of basic grid types



```
% Rectillinear grid
dx = 1−0.5*cos((−1:0.1:1)*pi);
x = −1.15+0.1*cumsum(dx);
G = tensorGrid(x, sqrt(0:0.05:1));
plotGrid(G);
```



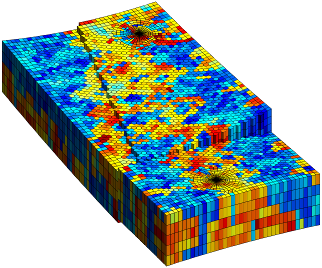
```
% Extrude a standard MATLAB dataset
load seamount
g = triangleGrid([x(:) y(:)]);
P = pebi(g);
V = makeLayeredGrid(P, 5);
plotGrid(V), view(−40, 60), axis off
```



```
% Make and read a simple Eclipse input file
grdecl = simpleGrdecl([20, 10, 5], 0.12)
G = processGRDECL(grdecl);
plotGrid(G,'FaceAlpha',0.8);
plotFaces(G,find(G.faces.tag>0), 'FaceColor','red');
view(40,40), axis off
```

Grids in MRST

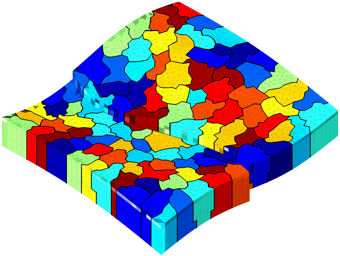
Examples of flexible gridding strategies



Hybrid grid – areal grid consisting of the following components

- radially refined grid at wells
- Cartesian along boundary
- hexahedral in interior
- polyhedral transition cells

extruded to 3D along vertical lines



Hierarchical grid

- triangular cells adapted to a curved line
- extruded to 3D, with throw along curved surface
- coarse grid constructed by collecting cells from fine grid

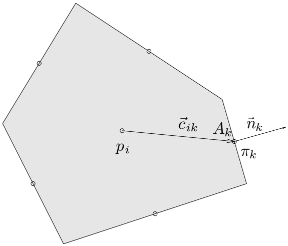
Discretization of flow equation

General family of conservative methods for: $\nabla \cdot \vec{v} = q, \quad \vec{v} = -\mathbf{K}\nabla p$

Basic formulation

$$\boldsymbol{u}_i = \boldsymbol{T}_i(\boldsymbol{e}_i p_i - \boldsymbol{\pi}_i), \qquad \boldsymbol{e}_i = (1, \ldots, 1)^\mathsf{T}$$

- p_i – the pressure at the center of cell i
- $\boldsymbol{u}_i$ – the vector of outward face fluxes
- $\boldsymbol{\pi}_i$ – the vector of face pressures
- $\boldsymbol{T}_i$ – the one-sided transmissibilities



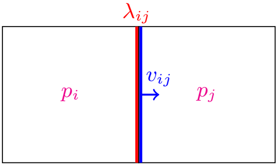
Special cases:

- The standard two-point method: $\boldsymbol{T}_{ii} = \vec{n}_i \cdot \boldsymbol{K}\vec{c}_i / |\vec{c}_i|^2$
- Multipoint flux-approximation methods (MPFA)
- Mixed finite-element methods
- Mimetic methods

Discretization of flow equation

Linear system: mixed hybrid form

$$\begin{bmatrix} B & C & D \\ C^T & 0 & 0 \\ D^T & 0 & 0 \end{bmatrix} \begin{bmatrix} v \\ -p \\ \pi \end{bmatrix} = \begin{bmatrix} 0 \\ g \\ 0 \end{bmatrix},$$



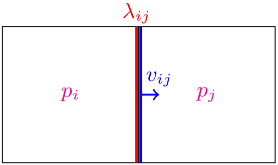
B defines an inner product. The matrix blocks read,

$$b_{ij} = \int_\Omega \psi_i \boldsymbol{T}^{-1} \psi_j \, dx, \quad c_{ik} = \int_\Omega \phi_k \nabla \cdot \psi_i \, dx, \quad d_{ik} = \int_{\partial\Omega} |\psi_i \cdot n_k| \, dx$$

Discretization of flow equation

Linear system: mixed hybrid form

$$\begin{bmatrix} B & C & D \\ C^T & 0 & 0 \\ D^T & 0 & 0 \end{bmatrix} \begin{bmatrix} v \\ -p \\ \pi \end{bmatrix} = \begin{bmatrix} 0 \\ g \\ 0 \end{bmatrix},$$



B defines an inner product. The matrix blocks read,

$$b_{ij} = \int_\Omega \psi_i \, \boldsymbol{T}^{-1} \psi_j \, dx, \quad c_{ik} = \int_\Omega \phi_k \nabla \cdot \psi_i \, dx, \quad d_{ik} = \int_{\partial\Omega} |\psi_i \cdot n_k| \, dx$$

Positive-definite system obtained by a Schur-complement reduction

$$\big(D^T B^{-1} D - F^T L^{-1} F\big)\pi = F^T L^{-1} g,$$
$$F = C^T B^{-1} D, \quad L = C^T B^{-1} C.$$

Reconstruct cell pressures and fluxes by back-substition,

$$Lp = q + F^T \pi, \qquad Bv = Cp - D\pi.$$

Discretization of flow equation

Herein: a mimetic method, Brezzi *et al.*, 2005

$$\boldsymbol{M}\boldsymbol{u} = (\boldsymbol{e}p - \boldsymbol{\pi}) \quad \longleftrightarrow \quad \boldsymbol{u} = \boldsymbol{T}(\boldsymbol{e}p - \boldsymbol{\pi})$$

Requiring exact solution of linear flow ($p = \boldsymbol{x}^\mathsf{T}\boldsymbol{a} + k$):

$$\boldsymbol{M}\boldsymbol{N}\boldsymbol{K} = \boldsymbol{C} \qquad \boldsymbol{N}\boldsymbol{K} = \boldsymbol{T}\boldsymbol{C}$$

$\boldsymbol{C}$ – vectors from cell to face centroids. $\boldsymbol{N}$ – area-weighted normal vectors

Family of schemes (given by explicit formulas):

$$\boldsymbol{M} = \frac{1}{|\Omega_i|}\boldsymbol{C}\boldsymbol{K}^{-1}\boldsymbol{C}^\mathsf{T} + \boldsymbol{Q}_N^{\perp\,\mathsf{T}}\boldsymbol{S}_M\boldsymbol{Q}_N^{\perp}$$

$$\boldsymbol{T} = \frac{1}{|\Omega_i|}\boldsymbol{N}\boldsymbol{K}\boldsymbol{N}^\mathsf{T} + \boldsymbol{Q}_C^{\perp\,\mathsf{T}}\boldsymbol{S}\boldsymbol{Q}_C^{\perp}$$

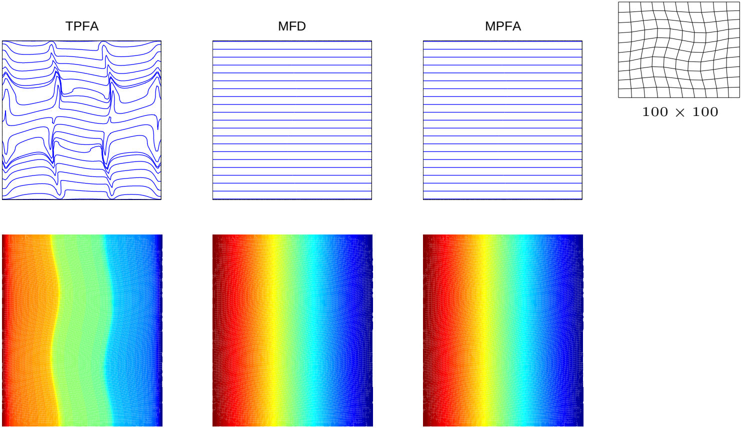
$\boldsymbol{Q}_N^{\perp}$ is an orthonormal basis for the null space of $\boldsymbol{N}^\mathsf{T}$, and $\boldsymbol{S}_M$ is any positive definite matrix. Herein, we use null-space projection

$$\boldsymbol{P}_N^{\perp} = \boldsymbol{Q}_N^{\perp}\boldsymbol{S}_M\boldsymbol{Q}_N^{\perp} = \boldsymbol{I} - \boldsymbol{Q}_N\boldsymbol{Q}_N^\mathsf{T}$$

Discretization of flow equation

Example: grid orientation effects

Homogeneous domain with Dirichlet boundary conditions (left,right) and no-flow conditions (top, bottom) computed with three different pressure solvers in MRST.

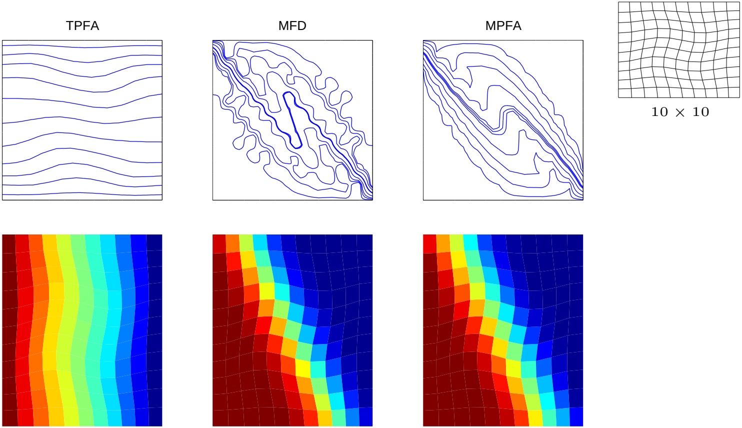


Homogeneous permeability with anisotropy ratio $1 : 1000$ aligned with the grid.

Discretization of flow equation

Example: lack of monotonicity

Homogeneous domain with Dirichlet boundary conditions (left,right) and no-flow conditions (top, bottom) computed with three different pressure solvers in MRST.



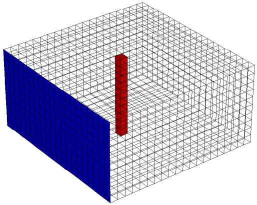
Homogeneous permeability with anisotropy ratio $1 : 1000$ rotated by $\pi/6$.

$$\nabla \cdot \vec{v} = q, \qquad \vec{v} = -\frac{\mathbf{K}}{\mu}\left[\nabla p + \rho g \nabla z\right]$$

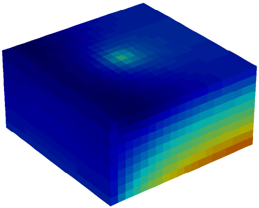
Vertical well and Dirichlet boundary

```
% Grid and rock parameters
nx = 20; ny = 20; nz = 10;
G = computeGeometry(cartGrid([nx, ny, nz]));
rock.perm = repmat(100 * milli*darcy, [G.cells.num, 1]);
fluid = initSingleFluid('mu', 1*centi*poise, ...
                        'rho', 1014*kilogram/meter^3);
gravity reset on

% Fluid sources and boundary conditions
c   = (nx/2*ny+nx/2 : nx*ny : nx*ny*nz) .';
src = addSource([], c, ones(size(c)) ./ day());
bc = pside([], G, 'LEFT', 10*barsa());

% Construct components for mimetic system
S = computeMimeticIP(G, rock, 'Verbose', true);

% Solve the system and convert to bars
rSol = initResSol(G, 0);
rSol = solveIncompFlow(rSol,G,S,fluid,'src',src,'bc',bc);
p = convertTo(rSol.pressure(1:G.cells.num), barsa() );
```

From tutorial: `simpleSRCandBC.m`



Source term and boundary condition



Pressure distribution

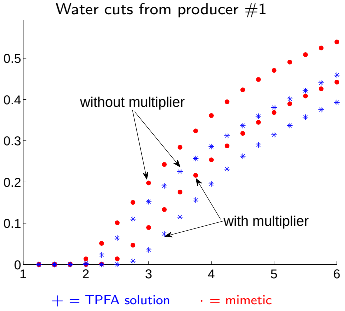
Application examples

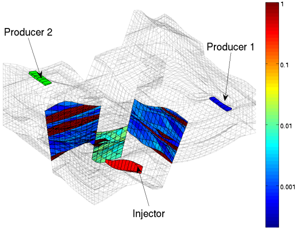
Modelling faults in consistent schemes

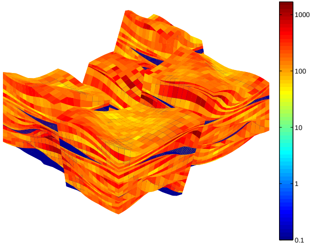
Faults modelled as internal boundaries, with internal jump conditions

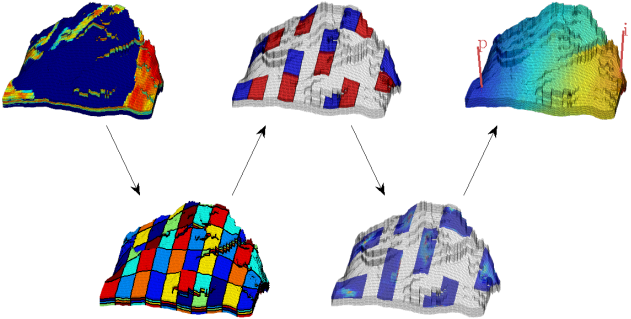
$$u_f^{\pm} = T_f(\pi_f^{\mp} - \pi_f^{\pm})$$

Gives an extended hybrid system. In addition, method to convert TPFA multipliers to fault transmissibility T_f









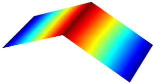
Key idea of multiscale methods:

Local decomposition

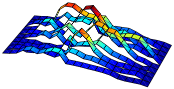
- $p(\vec{x}) = \sum_{\Omega_i} p_i \phi_i(\vec{x})$
- $\vec{v}(\vec{x}) = \sum_{\Gamma_{ij}} v_{ij} \vec{\psi}_{ij}(\vec{x})$

Ω_i – coarse grid block. $\Gamma_{ij} = \partial\Omega_i \cap \partial\Omega_j$

Multiscale basis functions:



homogeneous (RT0)



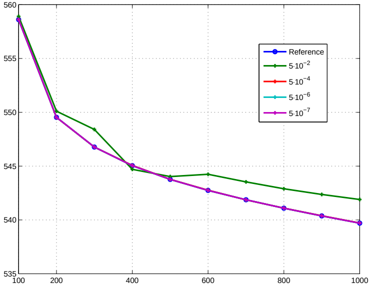
heterogeneous

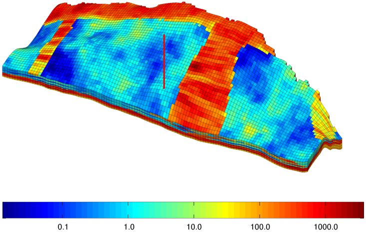
Application examples

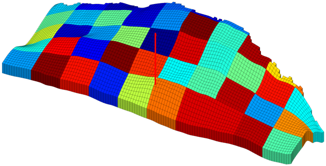
Multiscale module / black-oil module: primary production

- Shallow-marine reservoir (realization from SAIGUP)
- Model size: $40 \times 120 \times 20$
- Initially filled with gas, 200 bar
- Single producer, bhp=150 bar
- Multiscale solution for different tolerences compared with fine-scale reference solution.

Rate in well perforation (m^3/day)







Application examples

Coarsening module: (flow-based) coarsening by amalgamation

Amalgamation of cells

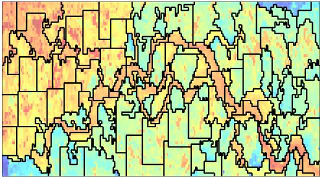
- flow-adapted grids
- simple and flexible coarsening
- adaptive gridding schemes
- efficient model reduction

Algorithmic primitives working on a partition vector

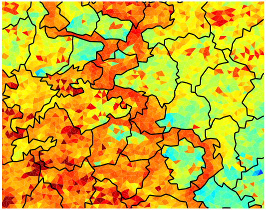
- segment (flow) indicator into bins
- merge small cells
- refine large cells
- intersect partitions
- sanity checks to ensure connected blocks, etc

$\longrightarrow$ great flexibility

Cartesian grid:



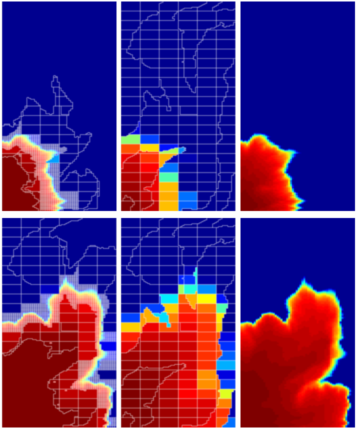
Triangular grids:



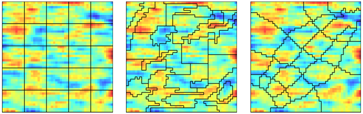
Application examples

Coarsening module: (flow-based) coarsening by amalgamation

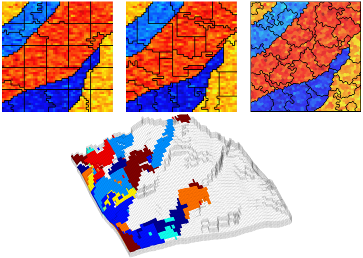
Dynamic adaption



Different partitioning:



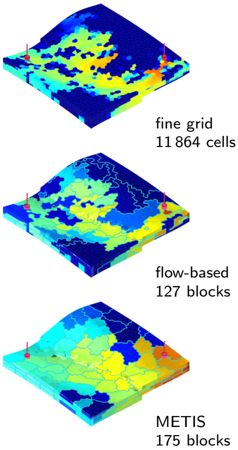
Adapting to geology



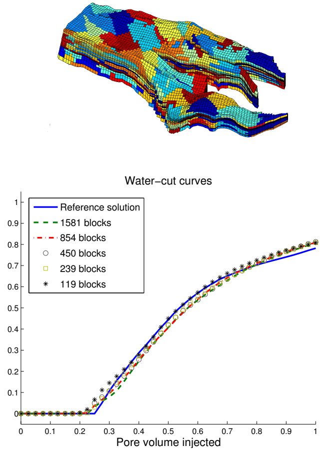
Application examples

Coarsening module: (flow-based) coarsening by amalgamation

Flow-adapted vs METIS:



Model reduction of real-field model:

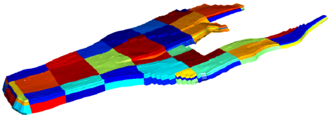


Application examples

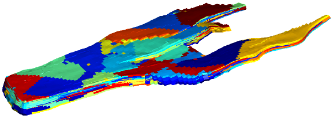
Upcoming adjoint module: production optimization

Specialized simulator: using different grids for pressure and transport

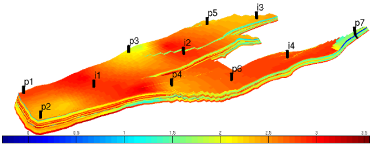
Multiscale pressure solver:



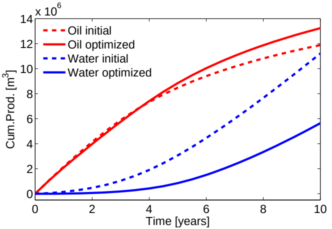
Transport on flow-adapted grid:



Water-flood optimization:



Reservoir geometry from a Norwegian Sea field

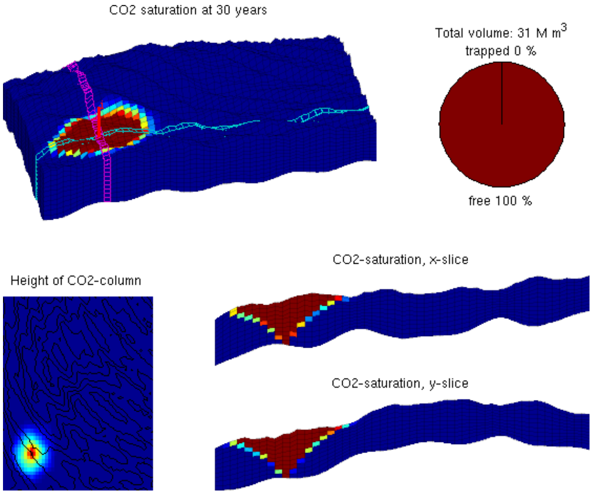


Forward simulations:

44 927 cells, 20 time steps, $<$ 5 sec in Matlab

Application examples

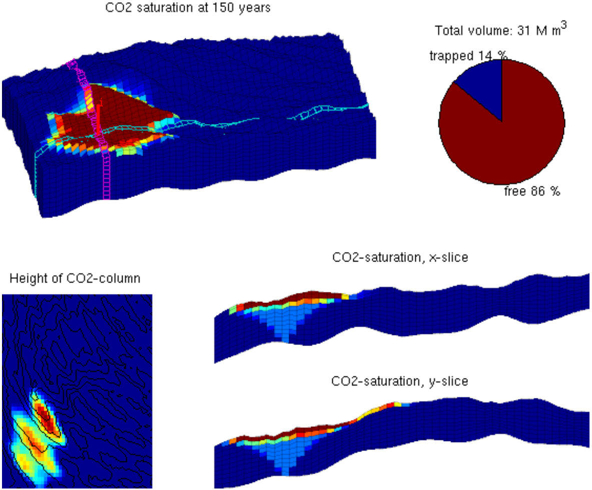
Vertically integrated module: educational and research tool



Vertically integrated module: part of a forthcoming numerical CO_2 laboratory

Application examples

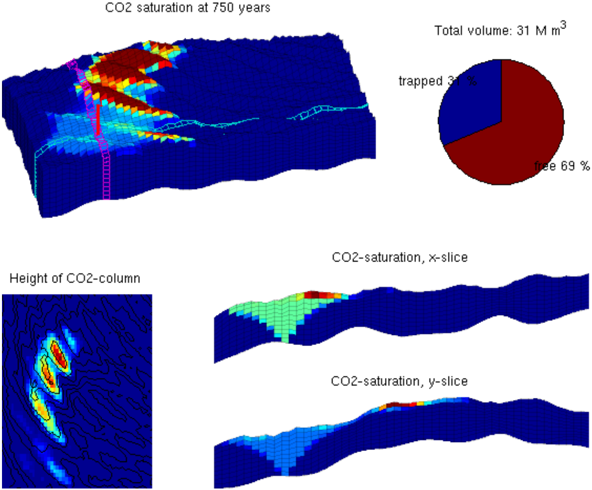
Vertically integrated module: educational and research tool



Vertically integrated module: part of a forthcoming numerical CO_2 laboratory

Application examples

Vertically integrated module: educational and research tool



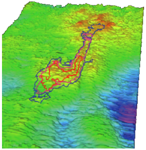
Vertically integrated module: part of a forthcoming numerical CO_2 laboratory

Application examples

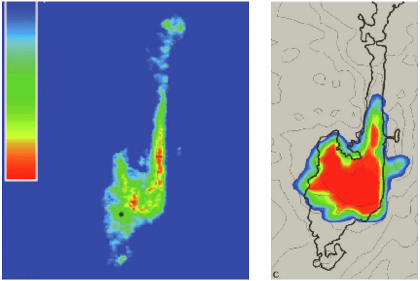
Vertically integrated module: CO_2 migration at Sleipner

3D model made by Statoil to study CO_2 in the top layer of Utsira.
MRST used to study physical assumptions and numerical methods:

- Simple to modify the code
- Flexible and simple upscaling (homogeneous model)
- Fast response time, simulation 2–10 min in Matlab



Seismic data (2006) / 3D simulation (tough2)



Fra: Chadwick, Noy, Arts & Eiken: Latest time-lapse seismic data from Sleipner yield new insights into CO_2 plume development, Energy Procedia (2009), 2103–2110.

VE simulation