Discretisation on Complex Grids – Open Source MATLAB Implementation

Knut–Andreas Lie Stein Krogstad Ingeborg S. Ligaarden Jostein R. Natvig
Halvor Møll Nilsen Bård Skaflestad*

30th June 2010

Abstract

Accurate geological modelling of features such as faults, fractures or erosion requires grids that are flexible with respect to geometry. Such grids generally contain polyhedral cells and complex grid cell connectivities. The grid representation for polyhedral grids in turn affects the efficient implementation of numerical methods for subsurface flow simulations.

Methods based on two-point flux approximations are known not to converge on grids that are not K orthogonal. In recent years, there has been significant research into mixed, multipoint, and mimetic discretisation methods that are all consistent and convergent. Furthermore, so-called multiscale methods have lately received a lot of attention.

In this paper we consider a MATLAB® implementation of consistent and convergent methods on unstructured, polyhedral grids. The main emphasis is put on flexibility and efficiency with respect to different grid formats, and in particular hierarchical grids used in multiscale methods. Moreover, we discuss how generic implementations of various popular methods for pressure and transport ease the study and development of advanced techniques such as multiscale methods, flow-based gridding, adjoint sensitivity analysis, and applications such as optimal control or well placement.

Introduction

Reliable computer modelling of subsurface flow is much needed to overcome important challenges such as sustainable use and management of the earth's groundwater systems, geological storage of CO_2 to mitigate the anthropological increases in the carbon content of the atmosphere, and optimal utilisation of hydrocarbon reservoirs. Indeed, the need for tools that help us understand flow processes in the subsurface is probably greater than ever, and increasing. More than fifty years of prior research in this area has led to some degree of agreement in terms of how subsurface flow processes can be modelled adequately with numerical simulation technology.

To describe the subsurface flow processes mathematically, two types of models are needed. First, one needs a mathematical model that describes how fluids flow in a porous medium. These models are typically given as a set of partial differential equations describing the mass-conservation of fluid phases, accompanied by a suitable set of constitutive relations. Second, one needs a geological model that describes the given porous rock formation (the reservoir). The geological model is realised as a grid populated with petrophysical or hydrological properties that are used as input to the flow model, and together they make up the reservoir simulation model. The geological model must also describe the geometry of the reservoir rock and in particular model geological horizons and major faults. This requires grids that are flexible with respect to geometry (and topology). Stratigraphic grids have been popular for many years and are the current industry standard. These grids are formed by extruding areal grids defined along geological surfaces to form volumetric descriptions. However, more complex methods based on unstructured grids are gaining in popularity as a means to modelling complex fault systems, horizontal and multilateral wells, etc. In either case, grids representing realistic reservoirs generally contain polyhedral cells and complex grid cell connectivities. The grid representation for polyhedral grids in turn affects the efficient implementation of numerical methods for subsurface flow simulations.

The industry-standard for discretising flow equations is the two-point flux-approximation method, which for a 2D Cartesian grid corresponds to a standard five-point scheme for the elliptic Poisson equation. Although widely used, this method is known not to converge on grids that are not $\mathbf{K}$-orthogonal. In recent years, there has been significant research into mixed (Brezzi and Fortin, 1991), multipoint (Aavatsmark, 2002), and mimetic (Brezzi et al., 2005) discretisation methods that are all consistent and convergent on rougher grids. Herein, we will focus on low-order, cell-centred methods that do not require specific reference elements and thus can be applied to grids with general polygonal and polyhedral cells.

Another major research challenge is the gap between simulation capabilities and the level of detail available in current geological models. Despite an astonishing increase in computer power, and intensive research on computation techniques, commercial reservoir simulators can seldom run simulations directly on highly resolved geological grid models that may contain from one to a hundred million cells. Instead, coarse-grid models with grid-blocks that are typically ten to a thousand times larger are built using some kind of upscaling of the geophysical parameters (Farmer, 2002; Durlofsky, 2005). How one should perform this upscaling is not trivial. In fact, upscaling has been, and probably still is, one of the most active research areas in the oil industry. Lately, however, so-called multiscale methods (Hou and Wu, 1997; Efendiev and Hou, 2009; Jenny et al., 2003) have received a lot of attention. In these methods, coarsening and upscaling needed to reduce the number of degrees of freedom to a level that is sufficient to resolve flow physics and satisfy requirements on computational costs is done implicitly by the simulation method.

A major goal of the activities in our research group is to develop efficient simulation methodologies based on accurate and robust discretisation methods; in particular, we have focused on developing multiscale methods. To this end, we need a toolbox for rapid prototyping of new ideas that enables us to easily test the new implementations on a wide range of models, from small and highly idealised grid models to large models with industry-standard complexity. When developing new computational methodologies, flexibility and low development time is more important than high code efficiency, which will typically only be fully achieved after the experimental programming is completed and ideas have been thoroughly tested. For a number of years, we have therefore divided our code development in two parts: For prototyping and testing of new ideas, we have used MATLAB, whereas solvers aimed at high computational performance have been developed in a compiled language (i.e., using FORTRAN, C, or generic programming in C++).

This has resulted in a comprehensive set of routines and data structures for reading, representing, processing, and visualising unstructured grids, with particular emphasis on the corner-point format used within the petroleum industry and hierarchical grids used in multiscale methods. To enable other researchers to benefit from out efforts, these routines have been gathered in the MATLAB Reservoir Simulation Toolbox (MRST), which is released under the GNU General Public License (GPL). The first releases are geared towards single- and two-phase flow and contain a set of mimetic and multiscale flow solvers and a few simple transport solvers capable of handling general unstructured, polyhedral grids.

The main purpose of this paper is to present MRST and demonstrate its flexibility and efficiency with respect to different grid formats, and in particular hierarchical grids used in multiscale methods. Moreover, we discuss how generic implementations of various popular methods for pressure and transport ease the study and development of advanced techniques such as multiscale methods, flow-based gridding, adjoint sensitivity analysis, and applications such as optimal control or well placement.

MATLAB Reservoir Simulation Toolbox

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 (our in-house development version also has support for compressible black-oil fluids), very simplified geostatistical routines
- routines for setting and manipulating boundary conditions, sources/sinks, and well models
- structure for reservoir state (pressure, fluxes, saturations, compositions, ...)
- visualisation routines for scalar cell and face data

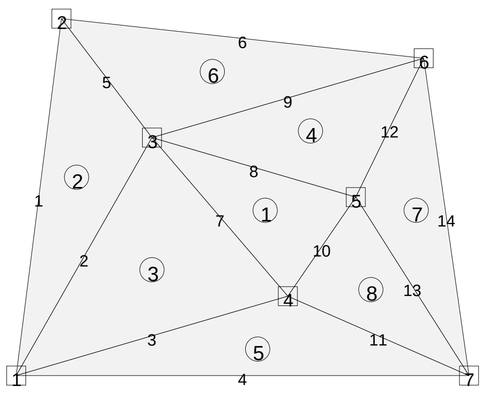
Additionally, the toolbox contains several flow and transport solvers which may be readily combined using an operator splitting framework. In particular, we provide an implementation of the multiscale mixed finite-element method (Aarnes et al., 2008), working on unstructured, polyhedral grids.

The toolbox assumes that all physical quantities are given in a consistent system of units, preferably the strict SI system. However, to assist the user, MRST provides conversion to/from common units in reservoir description.

We will now go into more details about some of the components outlined above. The interested reader should also review the tutorials included in the current release.

Grids

A key requirement for MRST is that it should support a large variety of grid types. To avoid having a large number of different and potentially incompatible grid representations, all grids in MRST are



```
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
    :    :             :    :
```

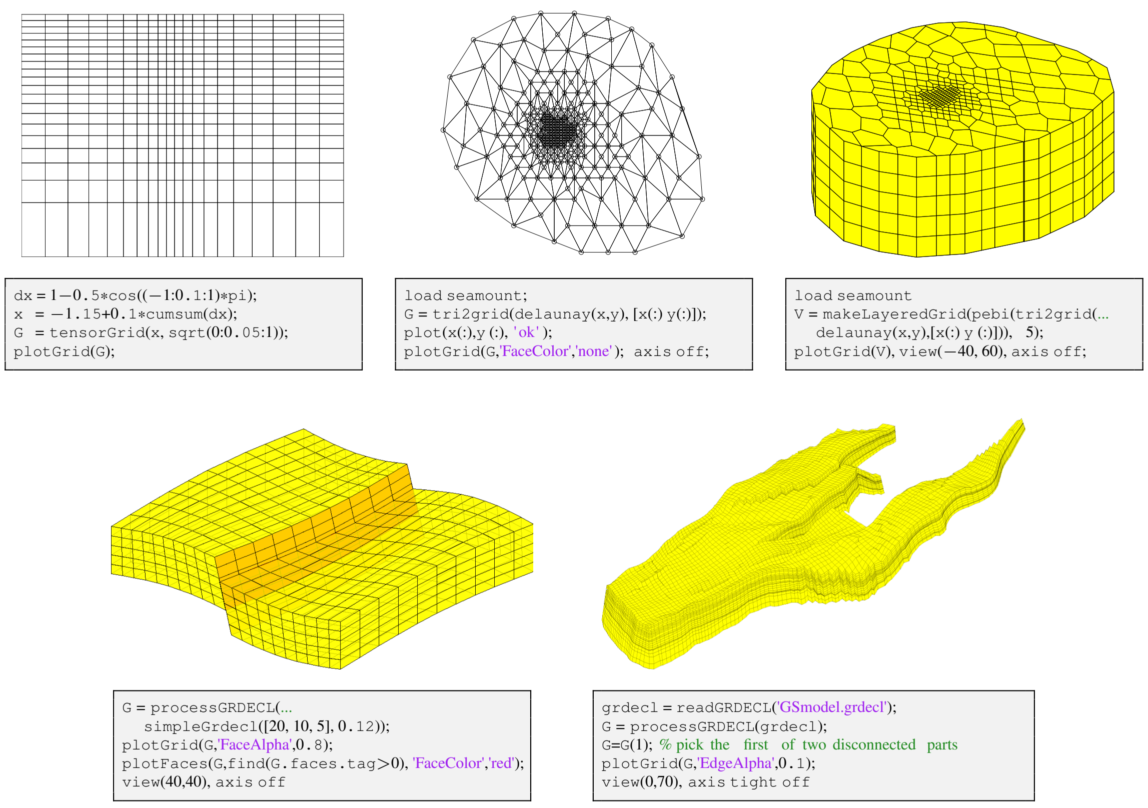
***Figure 1** Illustration of the `cell` and `faces` fields of the grid structure: cell numbers are marked by circles, node numbers by squares, and face numbers have no marker.*

assumed to consist of a set of non-overlapping polyhedral cells, where each cell can have a varying number of planar faces that match the faces of the cell's neighbours. Grids with non-matching faces, e.g., corner-point and other extruded grids, are therefore converted into matching grids by splitting non-matching faces into a set of matching (sub)faces. All grids are stored in a general format in which we explicitly represent cells, faces, and vertexes and the connections between cells and faces. Hence, we have sacrificed some of the efficiency attainable by exploiting special structures in a particular grid type for the sake of generality and flexibility.

The grid structure in MRST contains three fields—`cells`, `faces`, and `nodes`—that specify individual properties for each individual cell/face/vertex in the grid. The `nodes` structure is simple, it contains the number of nodes N_n and an $N_n \times d$ array of physical nodal coordinates in $\mathbb{R}^d$. The `cells` structure contains the number of cells N_c, an array `cells.faces` giving the global faces connected to a given cell, and an indirection map into `cells.faces` that gives the number of faces per cell. The `cells.faces` array has $n_f \times 2$ elements defined so that if `cells.faces`(i,1)==`j`, then global face `cells.faces`(i,2) is connected to global cell number j. To conserve memory, only the last column is stored, whereas the first column can be derived from a run-length encoding of the indirection map. The `cells` structure may optionally contain an array that maps internal cell indexes to external cell indexes, which is useful e.g., if the model contains inactive cells. Likewise, the `faces` structure contains the number of global faces, an array `faces.nodes` of vertexes connected to each cell, an indirection map, and an array `neighbors` giving neighbouring information (face i is shared by global cells `neighbors`(i,1) and `neighbors`(i,2)). In addition, the grid may contain a label `type` which records the grid construction history. Finally, grids supporting an underlying logical Cartesian structure also include the field `cartDims`. The grid structure is illustrated in Figure 1 for a triangular grid with eight cells.

MRST contains several grid-factory routines for creating structured grids, including regular Cartesian, rectilinear, and curvilinear grids, as well as unstructured grids, including Delaunay triangulations and Voronoi grids, and 3D grids created by extrusion of 2D shapes. Most important, however, is the support for the industry-standard corner-point grids given by the Eclipse input deck. In Figure 2 we show several examples of grids and the commands necessary to create and display them.

As we will see below, specific discretisation schemes may require other properties not supported by our basic grid class: cell volumes, cell centroids, face areas, face normals, and face centroids. Although these properties can be computed from the geometry (and topology) on the fly, it is often useful to precompute and include them explicitly in the grid structure `G`. This is done by calling the generic routine `G=computeGeometry(G)`.



```
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);
```

```
load seamount;
G = tri2grid(delaunay(x,y), [x(:) y(:)]);
plot(x(:),y(:), 'ok');
plotGrid(G,'FaceColor','none');  axis off;
```

```
load seamount
V = makeLayeredGrid(pebi(tri2grid(...
   delaunay(x,y),[x(:) y(:)])),  5);
plotGrid(V), view(-40, 60), axis off;
```

```
G = processGRDECL(...
   simpleGrdecl([20, 10, 5], 0.12));
plotGrid(G,'FaceAlpha',0.8);
plotFaces(G,find(G.faces.tag>0), 'FaceColor','red');
view(40,40), axis off
```

```
grdecl = readGRDECL('GSmodel.grdecl');
G = processGRDECL(grdecl);
G=G(1); % pick the  first  of two disconnected  parts
plotGrid(G,'EdgeAlpha',0.1);
view(0,70), axis tight off
```

Figure 2 *Examples of grids and the MRST statements necessary to create them. The data file* `seamount.mat` *is distributed with* MATLAB*; the file* `GSmodel.grdecl` *is not yet publicly available.*

Petrophysical Parameters

All flow and transport solvers in MRST assume that the rock parameters are represented as fields in a structure. Our naming convention is that this structure is called `rock`, but this is not a requirement. The fields for porosity and permeability, however, must be called `poro` and `perm`, respectively. The porosity field `rock.poro` is a vector with one value for each active cell in the corresponding grid model. The permeability field `rock.perm` can either contain a single column for an isotropic permeability, two or three columns for a diagonal permeability (in two and three spatial dimensions, respectively), or six columns for a symmetric, full tensor permeability. In the latter case, cell number i has the permeability tensor

$$\mathbf{K}_i = \begin{bmatrix} K_1(i) & K_2(i) \\ K_2(i) & K_3(i) \end{bmatrix}, \qquad \mathbf{K}_i = \begin{bmatrix} K_1(i) & K_2(i) & K_3(i) \\ K_2(i) & K_4(i) & K_5(i) \\ K_3(i) & K_5(i) & K_6(i) \end{bmatrix},$$

where $K_j(i)$ is the entry in column j and row i of `rock.perm`. Full-tensor, non-symmetric permeabilities are currently not supported in MRST. In addition to porosity and permeability, MRST supports a field called `ntg` that represents the net-to-gross ratio and consists of either a scalar or a single column with one value per active cell.

Given the difficulty of measuring rock properties, it is common to use geostatistical methods to make realisations of porosity and permeability. MRST contains two very simplified methods for generating geostatistical realisations. As a simple approximation to a Gaussian field, we generate a field of independent, normally distributed variables and convolve it with a Gaussian kernel. This method is used in two different routines, `gaussianField` and `logNormLayers` that are illustrated in the following examples.

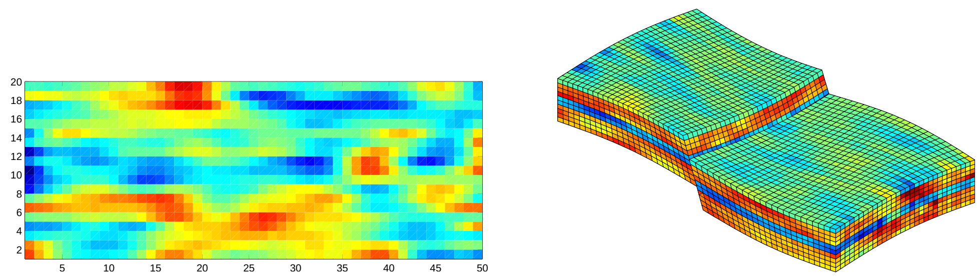


Figure 3 *Two examples of MRST's simplified geostatistics. The left plot shows a 50×20 porosity field generated as a Gaussian field with a larger filter size in the x-direction than in the y-direction. The right plot shows a stratigraphic grid with a single fault and four geological layers, each with a log-normal permeability distribution.*

Example 1 (Random Petrophysical Variables) *First, we generate the porosity ϕ as a Gaussian field taking values in the interval $[0.2, 0.4]$. To get a crude approximation to the permeability-porosity relationship, we assume that our medium is made up of uniform spherical grains of diameter $d_p = 10\,\mu\text{m}$, for which the specific surface area is $A_v = 6/d_p$. Using the Carman–Kozeny relation, we can then calculate the isotropic permeability K from*

$$K = \frac{1}{72\tau} \frac{\phi^3 d_p^2}{(1-\phi)^2},$$

where we further assume that $\tau = 0.81$. Then, petrophysical parameters can be generated as follows (the resulting porosity field is shown in the left plot of Figure 3):

```
G = cartGrid([50 20]);
p = gaussianField(G.cartDims, [0.2 0.4], [11 3], 2.5); p = p(:);
rock.poro = p;
rock.perm = p.^3.*(1e−5)^2./(0.81*72*(1−p).^2);
```

Next, we will use the same Gaussian field methodology to generate layered realisations, for which the permeability in each geological layer is independent of the other layers and log-normally distributed. Each layer can be represented by several grid cells in the vertical direction. Here, we generate a stratigraphic grid with wavy geological faces and a single fault and specify four geological layers with mean values of $100\,\text{mD}$, $400\,\text{mD}$, $50\,\text{mD}$, and $350\,\text{mD}$ from top to bottom (stratigraphic grids are numbered from the top and downward)

```
G = processGRDECL(simpleGrdecl([50 30 10], 0.12));
K = logNormLayers(G.cartDims, [100 400 50 350], 'indices', [1 2 5 7 11]);
```

The layers are represented with one, three, two, and four grid cells, respectively, in the vertical direction. The resulting permeability is shown in the right plot of Figure 3.

Using smoothed Gaussian fields to generate random petrophysical variables is, of course, a gross simplification of geostatistics. For more realistic distributions of petrophysical parameters, the reader should consider using e.g., GSLIB (Deutsch and Journel, 1998) or commercial software for geological modelling.

Discretisation of Flow Equations

To keep technical details at a minimum, we will in the following consider a simplified set of single-phase flow equations,

$$\nabla \cdot \vec{v} = q, \qquad \vec{v} = -\mathbf{K}\nabla p, \qquad \text{in } \Omega. \tag{1}$$

Here $\vec{v}$ denotes the Darcy velocity, p pressure, and K permeability. All external boundaries $\partial\Omega$ are equipped with either prescribed pressure (Dirichlet) or prescribed flux (Neumann) boundary conditions. Let $\boldsymbol{u}_i$ be the vector of outward fluxes of the faces of Ω_i and let p_i denote the pressure at the cell centre and $\boldsymbol{\pi}_i$ the face pressures. Discretisation methods used for reservoir simulation are constructed to be locally conservative and exact for linear solutions. Such schemes can be written in a form that relates these three quantities through a matrix $\boldsymbol{T}_i$ of half-transmissibilities,

$$\boldsymbol{u}_i = \boldsymbol{T}_i(\boldsymbol{e}_i p_i - \boldsymbol{\pi}_i), \qquad \boldsymbol{e}_i = (1, \ldots, 1)^{\mathsf{T}}. \tag{2}$$

Examples include the two-point flux-approximation method (Aziz and Settari, 1979), the lowest-order mixed finite-element methods (Brezzi and Fortin, 1991), multipoint flux approximation schemes (Aavatsmark et al., 1994; Edwards and Rogers, 1994; Aavatsmark, 2002), and recently developed mimetic finite-difference methods (Brezzi et al., 2005). Two-point discretisations give diagonal transmissibility matrices and are not convergent for general grids. Mixed, multipoint, and mimetic methods lead to full matrices $\boldsymbol{T}_i$. To ease the presentation, we henceforth only consider schemes that may be written in hybridised mixed form, although MRST also supports mixed forms. Note that this restriction, which excludes some multipoint schemes, is only necessary to give a uniform formulation of the schemes below. The underlying principles may be applied to any reasonable scheme.

To be precise, we consider a standard two-point scheme with half-transmissibilities

$$T_{if} = \vec{c}_{if} \cdot \boldsymbol{K}\vec{n}_f A_f / |\vec{c}_{if}|^2, \tag{3}$$

where A_f is the face area, $\vec{n}_f$ is the normal of the face, and $\vec{c}_{if}$ is the vector from the centroid of cell i to the centroid of the fth face. In MRST, the basic example of a consistent and convergent method on general polyhedral grids is the mimetic method with half-transmissibilities (Aarnes et al., 2008),

$$\boldsymbol{T} = \boldsymbol{N}\boldsymbol{K}\boldsymbol{N}^{\mathsf{T}} + (6/d) \cdot \mathrm{tr}(\boldsymbol{K})\boldsymbol{A}(\boldsymbol{I} - \boldsymbol{Q}\boldsymbol{Q}^{\mathsf{T}})\boldsymbol{A}. \tag{4}$$

Here each row in $\boldsymbol{N}$ is the cell outer normal to each face, $\boldsymbol{A}$ is the diagonal matrix of face areas and the matrix $\boldsymbol{I} - \boldsymbol{Q}\boldsymbol{Q}^{\mathsf{T}}$ spans the null space of the matrix of centroid differences. Likewise, we have implemented the MPFA-O method using a local-flux mimetic formulation (Lipnikov et al., 2009) that fits into the same framework.

Augmenting (2) with flux and pressure continuity across cell faces, we get the following linear system

$$\begin{bmatrix} \boldsymbol{B} & \boldsymbol{C} & \boldsymbol{D} \\ \boldsymbol{C}^{\mathsf{T}} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{D}^{\mathsf{T}} & \boldsymbol{0} & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{u} \\ -\boldsymbol{p} \\ \boldsymbol{\pi} \end{bmatrix} = \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{q} \\ \boldsymbol{0} \end{bmatrix}, \tag{5}$$

where the first row in the block-matrix equation corresponds to (2) for all grid cells. Thus, $\boldsymbol{u}$ denotes the outward face fluxes ordered cell-wise (fluxes over interior faces and faults appear twice with opposite signs), $\boldsymbol{p}$ denotes the cell pressures, and $\boldsymbol{\pi}$ the face pressures. The matrices $\boldsymbol{B}$ and $\boldsymbol{C}$ are block diagonal with each block corresponding to a cell. For the two matrices, the i'th blocks are given as $\boldsymbol{T}_i^{-1}$ and $\boldsymbol{e}_i$, respectively. Similarly, each column of $\boldsymbol{D}$ corresponds to a unique face and has one (for boundary faces) or two (for interior faces) unit entries corresponding to the index(s) of the face in the cell-wise ordering. The hybrid system (5) can be solved using a Schur-complement method and MATLAB's standard linear solvers or third party linear system solver software such as AGMG (Notay, 2010).

Putting it all Together

In this section, we will go through a very simple example to give an overview of how to set up and use the mimetic pressure solver introduced in the previous section to solve the single-phase pressure

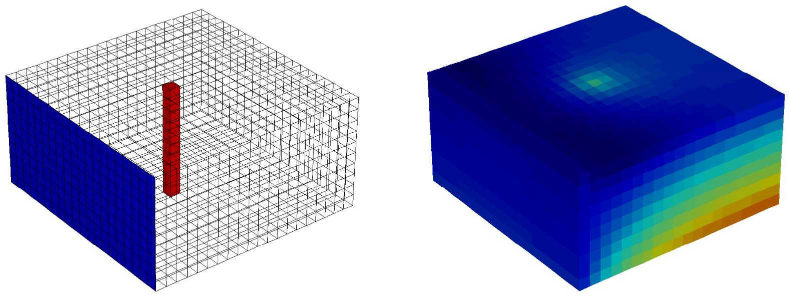


Figure 4 *Example of a simple flow driven by a column of source cells and a Dirichlet boundary condition. The left plot shows the model setup and the right plot the corresponding pressure solution.*

equation

$$\nabla \cdot \vec{v} = q, \qquad \vec{v} = -\frac{\mathbf{K}}{\mu}\big[\nabla p + \rho g \nabla z\big]. \tag{6}$$

First, we construct a Cartesian grid of size $n_x \times n_y \times n_z$ cells, where each cell has dimension $1 \times 1 \times 1\,\mathrm{m}$ and set an isotropic and homogeneous permeability of $100\,\mathrm{mD}$, a fluid viscosity of $1\,\mathrm{cP}$ and a fluid density of $1014\,\mathrm{kg/m^3}$.

```
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
```

The simplest way to model inflow or outflow from the reservoir is to use a fluid source/sink. Here, we specify a source with flux rate of $1\,\mathrm{m^3}$/day in each grid cell.

```
c   = (nx/2*ny+nx/2 : nx*ny : nx*ny*nz) .';
src = addSource([], c, ones(size(c)) ./ day());
```

Our flow solvers automatically assume no-flow conditions on all outer (and inner) boundaries; other types of boundary conditions need to be specified explicitly. To draw fluid out of the domain, we impose a Dirichlet boundary condition of $p = 10\,\mathrm{bar}$ at the global left-hand side of the model.

```
bc = pside([], G,  'LEFT', 10*barsa());
```

Here, the first argument has been left empty because this is the first boundary condition we prescribe. The left plot in Figure 4 shows the placement of boundary conditions and sources in the computational domain. Next, we construct the system components for the hybrid mimetic system (5) based on input grid and rock properties.

```
S = computeMimeticIP(G, rock, 'Verbose', true);
```

Rather than storing $\boldsymbol{B}$, we store its inverse $\boldsymbol{B}^{-1}$. Similarly, the $\boldsymbol{C}$ and $\boldsymbol{D}$ blocks are not represented in the S structure; they can easily be formed explicitly whenever needed, or their action can easily be computed.

Finally, we compute the solution to the flow equation. To this end, we must first create a state object that will be used to hold the solution and then pass this objects and the parameters to the incompressible flow solver.

```
rSol = initResSol(G, 0);
rSol = solveIncompFlow(rSol, G, S, fluid,'src', src, 'bc', bc);
p = convertTo(rSol.pressure(1:G.cells.num), barsa() );
```

Having computed the solution, we convert the result back to the unit bars. The right plot in Figure 4 shows the corresponding pressure distribution, where we clearly can see the effects of the boundary conditions, the source term, and gravity.

The same basic steps can be repeated on (almost) any type of grid; the only difference is placing the source terms and how to set the boundary conditions, which will typically be more complicated on a fully unstructured grid. We will come back with more examples later in the paper, but then we will not explicitly state all details of the corresponding MRST scripts. Before giving more examples, however, we will introduce the multiscale flow solver implemented in MRST.

Multiscale Flow Simulation

Multiscale flow solvers (Hou and Wu, 1997; Efendiev and Hou, 2009) can be seen as numerical methods that take a fine-scale model as input, but solve for a reduced set of unknowns defined on a coarse grid to produce a solution that has both coarse-scale and fine-scale resolution. A key characteristic with multiscale methods is that they incorporate fine-scale information into a set of coarse-scale equations in a way that is consistent with the local property of the differential operator. In an abstract formulation, a multiscale method uses two operators: a compression (or restriction) operator that takes information from the fine scale to the coarse scale, and a reconstruction (or prolongation) operator that takes information from the coarse scale to the fine scale. The compression operator is used to reduce the system of discretised flow equations on a fine grid to a system with significantly fewer unknowns defined on a coarse grid. Similarly, the reconstruction operator is used to construct an approximate fine-scale solution from the solution computed on the coarse scale. The reconstruction operators are typically computed numerically by solving a localised flow problem as in an upscaling method.

Different multiscale flow solvers are distinguished by how they define their degrees of freedom and the compression and reconstruction operators. In the multiscale finite-volume (MsFV) method (Jenny et al., 2003; Zhou and Tchelepi, 2008), the coarse-scale degrees-of-freedom are associated with pressure values at the vertexes of the coarse grid. The reconstruction operator is associated with pressure values and is defined by solving flow problems on a dual coarse grid. (In addition, the method needs to solve a local flow problem on the primal coarse grid to recover conservative fine-scale fluxes). In the multiscale mixed finite-element (MsMFE) method (Chen and Hou, 2003; Aarnes et al., 2008), the coarse-scale degrees-of-freedom are associated with faces in the coarse grid (coarse-grid fluxes) and the reconstruction operator is associated with the fluxes and is defined by solving flow problems on a primal coarse grid. In the following, we will present the MsMFE method in more detail. To this end, we will use a finite-element formulation, but the resulting discrete method will have all the characteristics of a (mass-conservative) finite-volume method.

The multiscale method implemented in MRST is based on a hierarchical two-level grid in which the blocks Ω_i in the coarse simulation grid consist of a connected set of cells from the underlying fine grid, on which the full heterogeneity is represented. In its simplest form, the approximation space consists of a constant approximation of the pressure inside each coarse block and a set of velocity basis functions associated with each interface between two coarse blocks. Consider two neighbouring blocks Ω_i and Ω_j, and let Ω_{ij} be a neighbourhood containing Ω_i and Ω_j. The basis functions $\vec{\psi}_{ij}$ are constructed by solving

$$\vec{\psi}_{ij} = -\mathbf{K}\nabla p_{ij}, \qquad \nabla \cdot \vec{\psi}_{ij} = \begin{cases} w_i(x), & \text{if } x \in \Omega_i, \\ -w_j(x), & \text{if } x \in \Omega_j, \\ 0, & \text{otherwise}, \end{cases} \tag{7}$$

in Ω_{ij} with $\vec{\psi}_{ij} \cdot \vec{n} = 0$ on $\partial\Omega_{ij}$. If $\Omega_{ij} \neq \Omega_i \cup \Omega_j$, we say that the basis function is computed using

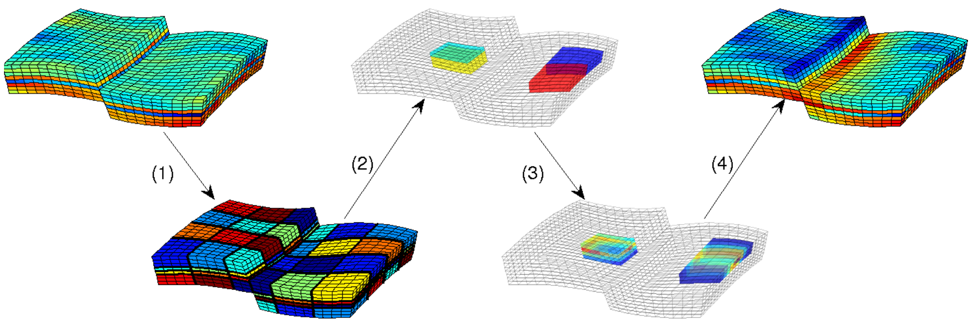


***Figure 5** Key steps of the multiscale method: (1) blocks in the coarse grid are defined as connected collections of cells from the fine grid; (2) a local flow problem is defined for all pairs of blocks sharing a common face; (3) the local flow problems are solved and the solutions are collected as basis functions (reconstruction operators); and (4) the global coarse-system* (8) *is assembled and solved, then a fine-scale solution can be reconstructed.*

overlap or oversampling. The purpose of the weighting function $w_i(x)$ is to distribute the divergence of the velocity, $\nabla \cdot \vec{v}$, over the block and produce a flow with unit flux over the interface $\partial\Omega_i \cap \partial\Omega_j$, and the function is therefore normalised such that its integral over Ω_i equals one. Alternatively, the method can be formulated on a single grid block Ω_i by specifying a flux distribution (with unit average) on one face and no-flow condition on the other faces, see (Aarnes et al., 2005) for more details. In either case, the multiscale basis functions—represented as vectors Ψ_{ij} of fluxes—are then collected as columns in a matrix $\boldsymbol{\Psi}$, which will be our reconstruction operator for fine-scale fluxes. To define the compression operator, we introduce two prolongation operators $\boldsymbol{\mathcal{I}}$ and $\boldsymbol{\mathcal{J}}$ from blocks to cells and from coarse interfaces to fine faces, respectively. The operator $\boldsymbol{\mathcal{I}}$ is defined such that element ij equals one if block number j contains cell number i and zero otherwise; $\boldsymbol{\mathcal{J}}$ is defined analogously. The transposed of these operators will thus correspond to the sum over all fine cells of a coarse block and all fine-cell faces that are part of the faces of the coarse blocks. Applying these compression operators and $\boldsymbol{\Psi}^\mathsf{T}$ to the fine-scale system, we obtain the following global coarse-scale system

$$\begin{bmatrix} \boldsymbol{\Psi}^\mathsf{T}\boldsymbol{B}\boldsymbol{\Psi} & \boldsymbol{\Psi}^\mathsf{T}\boldsymbol{C}\boldsymbol{\mathcal{I}} & \boldsymbol{\Psi}^\mathsf{T}\boldsymbol{D}\boldsymbol{\mathcal{J}} \\ \boldsymbol{\mathcal{I}}^\mathsf{T}\boldsymbol{C}^\mathsf{T}\boldsymbol{\Psi} & \mathbf{0} & \mathbf{0} \\ \boldsymbol{\mathcal{J}}^\mathsf{T}\boldsymbol{D}^\mathsf{T}\boldsymbol{\Psi} & \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{u}^c \\ -\boldsymbol{p}^c \\ \boldsymbol{\pi}^c \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \boldsymbol{\mathcal{I}}^\mathsf{T}\boldsymbol{q} \\ \mathbf{0} \end{bmatrix}. \qquad (8)$$

Once (8) is solved, the fine-scale fluxes can be obtained immediately as $\boldsymbol{v} = \boldsymbol{\Psi}\boldsymbol{u}^c$. The basic steps of the multiscale algorithm are summarised in Figure 5. More details about how to include wells in a appropriate way is given in (Skaflestad and Krogstad, 2008).

Having introduced the multiscale method, we should explain how to use it. To this end, we consider a simple example.

Example 2 (Log-Normal Layered Permeability) *In this example, we will revisit the setup from the previous section. However, we neglect gravity and instead of assuming a homogeneous permeability, we increase the number of cells in the vertical direction to 20 and impose a log-normal, layered permeability as shown in Figure 3, and use the layering from the permeability to determine the extent of the coarse blocks; a large number of numerical experiments have shown that the MsMFE method gives best resolution when the coarse blocks follow geological layers (Aarnes et al., 2008). In MRST, this amounts to:*

```
[K, L]     = logNormLayers([nx, ny, nz], [200 45 350 25 150 300], 'sigma',   1);
:
p = processPartition(G, partitionLayers(G, [Nx, Ny], L) );
:
plotCellData(G,mod(p,2));
outlineCoarseGrid(G,p,'LineWidth',3);
```

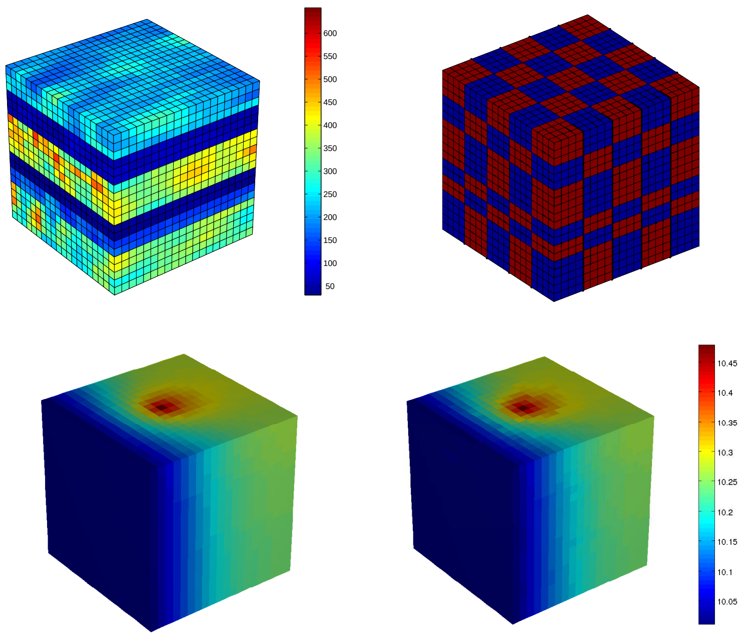


Figure 6 *Comparison of the MsMFE and fine-scale mimetic flow solvers for the setup from Figure 4. The top plots show the layered permeability field and the corresponding partitioning into coarse blocks. The lower plots show the fine-scale pressure solution (left) and the multiscale solution (right).*

The permeability and the coarse grid are shown in Figure 6. Having partitioned the grid, the next step is to build the grid structure for the coarse grid and generate and solve the coarse-grid system.

```
CG = generateCoarseGrid(G, p);
CS = generateCoarseSystem(G, rock, S, CG, ones([G.cells.num, 1]),'bc', bc, 'src', src);
xMs = solveIncompFlowMS(initResSol(G, 0.0), G, CG, p, S, CS, fluid,'src', src, 'bc', bc);
```

The multiscale pressure solution is compared to the corresponding fine-scale solution in Figure 6. The solutions appear to be quite close in the visual norm.

For single-phase problems, a multiscale flow solver without any form of parallelism will have a computational cost that is comparable to that of a fine-scale flow solver equipped with an efficient linear routine, i.e., MATLAB's built-in solvers for small problems and e.g., the AGMG solver (Notay, 2010) for large problems. The advantage of a multiscale solver comes first when considering multiphase flow problems, as we will discuss in the next section.

Two-Phase Flow

Two-phase incompressible flow of a wetting and non-wetting fluid can be described by the following system of equations:

$$\nabla \cdot \vec{v} = q, \qquad \vec{v} = -\mathbf{K}\big[\lambda \nabla p + (\lambda_w \rho_w + \lambda_n \rho_n) g \nabla z\big], \tag{9}$$

$$\phi \frac{\partial s_w}{\partial t} + \nabla \cdot \Big(f_w(s_w)\big[\vec{v} + \lambda_n(\rho_n - \rho_w) g \mathbf{K} \nabla z\big]\Big) = q_w. \tag{10}$$

Here, ρ_α denotes the density, λ_α the mobility, and $f_\alpha = \lambda_\alpha/\lambda$ the fractional flow of phase α, where $\lambda = \lambda_n + \lambda_w$ is the total mobility. The industry-standard approach is to use implicit discretisations and solve (9)–(10) as a fully-coupled system. In MRST, on the other hand, our goal has been to obtain maximum flexibility in combining different types of solvers. Hence, the toolbox assumes a sequential solution strategy: First, (9) is solved with fixed saturation values to provide pressure and fluxes, and then the fluxes are used to evolve the saturations according to (10). If necessary, the two steps can be iterated until convergence in a suitable norm.

All flow solvers in MRST are fully applicable to the two-phase flow equation (9), except for the MsMFE method, which does not support gravity in the current release of MRST. On the other hand, multiscale flow solvers are developed mainly to be efficient for multiphase problems, and in particular for two-phase incompressible equations, where the key to computational efficiency is reuse of previous computations. For a multiphase system, the basis functions will become time-dependent when K is replaced by $\lambda\mathsf{K}$ in (7), but this time-dependence is weak for an incompressible system, and the basis functions can therefore be computed initially and updated infrequently throughout a simulation. Solving the coarse-scale problem (8) is usually inexpensive compared to solving the corresponding fine-scale system or computing basis functions.

MRST supports two basic saturation solvers that both use a single-point upwind discretisation. Dropping subscripts to denote phases and assuming no gravity, they can be written in the form

$$s_i^{n+1} = s_i^n + \frac{\Delta t}{\phi_i|c_i|}\Big(\max(q_i,0) + f(s_i^m)\min(q_i,0)\Big) - \frac{\Delta t}{\phi_i|c_i|}\Big(\sum_j\big[f(s_i^m)\max(v_{ij},0) + f(s_j^m)\min(v_{ij},0)\big]\Big), \tag{11}$$

Here, s_i is the average saturation in grid cell c_i, v_{ij} denotes the flux over the face between cells i and j. For $m = n$, the scheme is explicit, whereas for $m = n + 1$, we obtain an implicit scheme that is solved by a Newton–Raphson method. For systems with gravity forces, MRST uses standard upstream mobility weighing; that is, the upwind direction is determined independently for each phase using the phase velocities $\vec{v}_\alpha$.

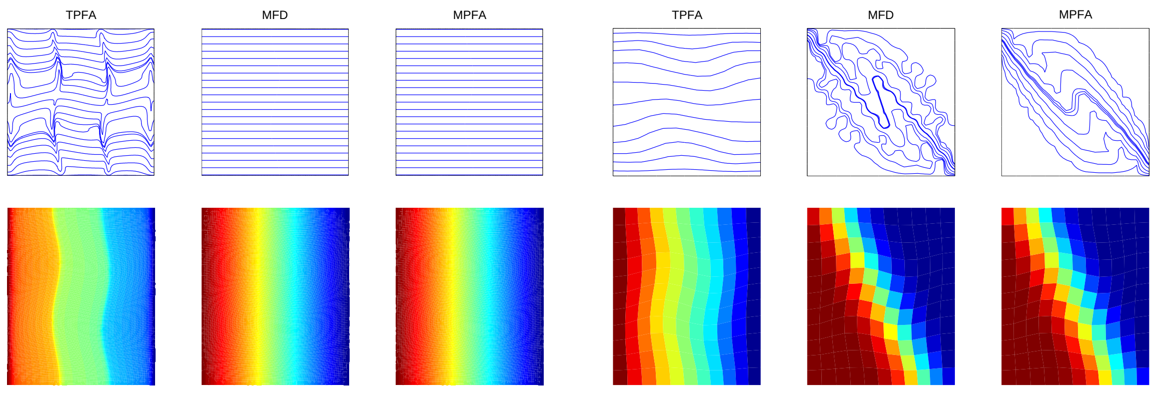
The observant reader may have noticed that the capillary pressure is missing in our two-phase model. In (9), capillary effects can be included by defining the pressure as the *global* pressure, $p = p_n - p_c$, where the so-called complementary pressure is defined through the relation $\nabla p_c = f_w \nabla(p_n - p_w)$.

So far, the paper has given a quick overview of the basic functionality and solvers in MRST Release 2010a. The next section will present more examples in which the solvers are applied to more advanced examples, but also examples of solvers and functionality that are not yet released publicly.

Advanced Examples

A main purpose of MRST is to provide support for different discretisation methods. As explained above, MRST currently has three different methods implemented for fully unstructured grids: a scheme based on two-point flux approximations (TPFA), a mimetic finite difference (MFD) method (Brezzi et al., 2005; Aarnes et al., 2008) and a multipoint flux approximation (MPFA) 'O'-scheme (Aavatsmark, 2002). It is well-known that two-point approximations are convergent only in the special case of K-orthogonal grids. On the other hand, mimetic finite-difference and multipoint schemes are constructed to be convergent for rough grids and full-tensor permeabilities. However, the latter schemes may suffer from pressure oscillations for full-tensor permeabilities and large anisotropy ratios and/or high aspect ratio grids. Our first example will use one of the example grids supplied with MRST to illustrate these two observations.

Example 3 (Grid-Orientation Effects and Monotonicity) *In this example, we will use a curvilinear grid obtained by a smooth mapping of a uniform Cartesian grid. To this end, we will use the MRST routine* `twister`*, which normalises all coordinates in a rectilinear grid to the interval* $[0, 1]$*, then perturbs all interior points by adding* $0.03\sin(\pi x)\sin(3\pi(y - 1/2))$ *to the* x*-coordinates and subtracting the same value from the* y *coordinate, before transforming back to the original domain. This creates a*



Grid-orientation effects   Effects of monotonicity

Figure 7 *Velocity field shown as streamlines and pressure field in a homogeneous domain with Dirichlet boundary conditions (left,right) and no-flow conditions (top, bottom) computed with three different pressure solvers in MRST. The permeability field is homogeneous with anisotropy ratio $1 : 1000$ aligned with the grid (left) and rotated by $\pi/6$ (right).*

non-orthogonal, but smoothly varying grid.

To demonstrate the difference in grid-orientation effects for the three flow solvers in MRST, we impose a permeability tensor with anisotropy ratio $1 : 1000$ aligned with the x-axis and compute the flow resulting from a a prescribe horizontal pressure drop and no-flow top and bottom boundaries. The left-hand plots in Figure 7 show pressure computed by the TPFA, MFD, and MPFA-O flow solvers on a 100×100 grid. To visualise the corresponding velocity field, we also show streamlines traced by MRST's implementation of Pollock's method. Whereas the MFD and MPFA-O schemes produce the expected result, the pressure and velocity solutions computed by TPFA show significant grid-orientation effects and are obviously wrong.

On the right in Figure 7, we have repeated the experiment with the tensor rotated by $\pi/6$ on a grid with 11×11 cells. Again, we see that the solution computed with the TPFA scheme is wrong. In this case, the pressure computed with the MFD and the MPFA schemes seems correct and has no discernible oscillations. However, the streamline distributions clearly show that the resulting velocity field is highly oscillatory.

Given the importance of grid geometry on the quality of numerical computations, it is crucial to have flexible tools that allow testing algorithms on many different types of grids. In the next example we highlight the ability to construct unstructured grids with complex connectivity and local refinement.

Example 4 (Flexible Gridding) *In this example we have constructed a constrained Voronoi grid from an initial distribution of points. With the built-in* MATLAB *command* `voronoin`*, it is possible to construct uniform grids from uniform distributions of points. In this example we extrude an unstructured areal grid with 4509 cells to a faulted model with 5 layers. The areal grid consist of three parts: A Cartesian grid at the outer boundary, a hexahedral grid in the interior, and a radial grid with exponential radial refinement around the wells. The radial grids have been refined to an inner radius of* $0.3\,\mathrm{m}$.

As in the standard corner-point grid format, the extrusion process is accomplished by assigning pillars to each node in the areal grid. Each grid cell in the extruded grid is given a set of z-coordinates for each of the nodes. The x and y coordinates can be computed from the pillars associated with the cell.

To illustrate the use of this grid, we have sampled permeability and porosity from the layers 40 through 44 of the 10th SPE Comparative Solution Project (Christie and Blunt, 2001). The permeability field

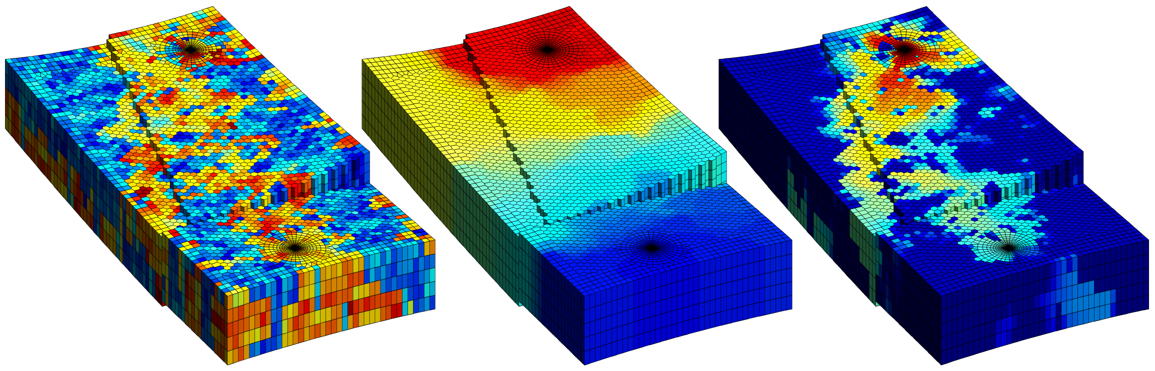


Figure 8 *Two-phase flow computed on a pillar grid constructed by extruding an areal grid consisting of structured and unstructured parts: a radially refined grid, a Cartesian grid along the outer boundary, a hexahedral grid in the interior and general polyhedral cells to glue everything together. The figure shows permeability (left), pressure field (middle) and water saturation after* 5000 *days.*

is shown on the left in Figure 8. The wells are modelled as Dirichlet boundaries with pressures set to $200\,\mathrm{bar}$ *and* $100\,\mathrm{bar}$*, respectively, and we compute the pressure and water saturation with a sequentially implicit scheme in time steps of* 50 *days using the mimetic flow solver. The oil-water viscosity ratio is* 10*. The pressure and water saturation is shown after* 5000 *days of injection in Figure 8.*

In the rest of this section, we will go through several examples taken from our ongoing research activities, in which MRST is used to simulate flow on polyhedral grids of varying complexity in 3D. By doing so, we hope to give the reader a taste of the utility of the toolbox. We start by a short example which demonstrates the applicability of the MsMFE method on unstructured grids in 3D. Although several of our papers have argued that the method can easily be formulated on fully unstructured grids in 3D, the following is the first example demonstrating the actual capability on grids that do not have an underlying logically Cartesian structure.

Example 5 (MsMFEM on Extruded Triangular and PEBI Grids) *The MsMFE method was previously demonstrated on Cartesian and logically Cartesian grids such as corner-point grids, see (Aarnes et al., 2006; Kippe et al., 2008; Aarnes et al., 2008). However, the method is equally feasible on general, unstructured grids provided there exists a suitable partitioning of the underlying (fine-scale) grid. Figure 9 demonstrates the solution of the single-phase flow problem* (1) *with isotropic, homogeneous* K *on a computational domain* Ω *that has been discretised into both triangular and PEBI cells. These cells are subsequently partitioned into a coarse grid by means of the well-known software package METIS (Karypis and Kumar, 1995). For simplicity, the PEBI grid was created as the Voronoi diagram of the triangular grid and hence the fault is more irregular on this grid. Despite the irregular block boundaries, the multiscale method is able to accurately capture the linear pressure drop from the west to the east boundaries.*

For multiphase problems, a multiscale flow solver must be accompanied by an efficient transport solver working on the coarse grid, on the underlying fine grid, or on some intermediate grid. On the fine grid, one can either use streamlines (Aarnes et al., 2005; Natvig et al., 2009) or an implicit finite-volume method with a localised nonlinear solver (Natvig and Lie, 2008a,b). On coarse and intermediate grids, where each grid block B_ℓ will typically consist of a connected collection of cells c_i from the fine grid, a simple and yet efficient approach is to use the following finite-volume method,

$$\begin{aligned} S_\ell^{n+1} = S_\ell^n + \frac{\Delta t}{\sum_{B_\ell} \phi_i |c_i|} \Big[&\max(q_\ell, 0) + f(S_\ell^{n+1}) \min(q_\ell, 0) \\ &- f(S_\ell^{n+1}) \sum_{\gamma_{ij} \subset \partial B_\ell} \max(v_{ij}, 0) + \sum_{k \neq \ell} \Big(f(S_k^{n+1}) \sum_{\gamma_{ij} \subset \Gamma_{k\ell}} \min(v_{ij}, 0) \Big) \Big]. \qquad (12) \end{aligned}$$

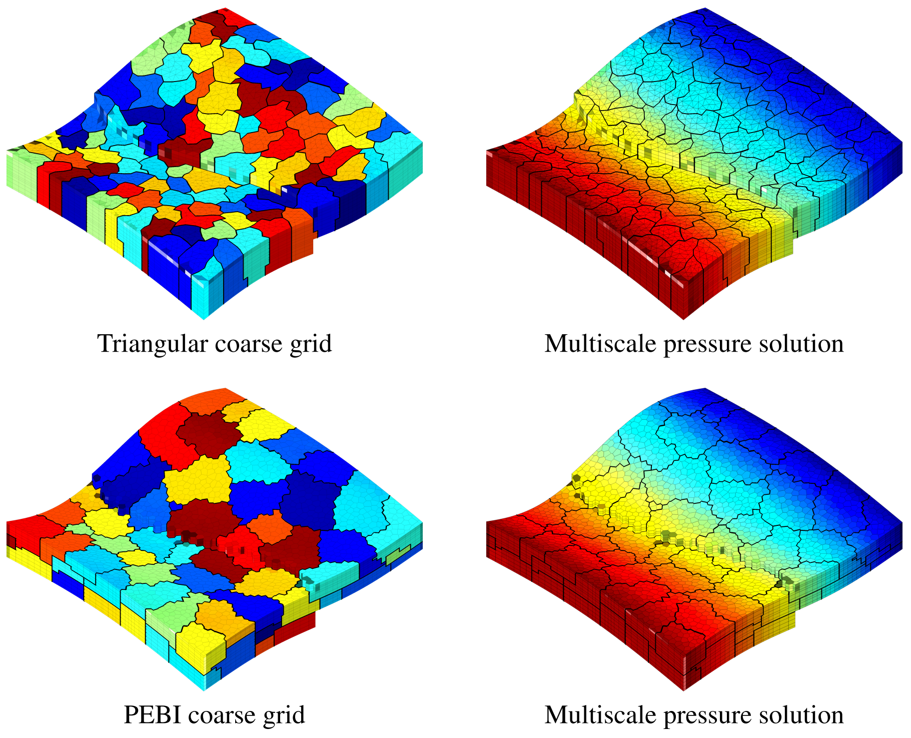


Figure 9 *Multiscale discretisation of a flow problem with linear pressure drop on a triangular and a PEBI grid.*

Here, S_ℓ denotes average saturation in block B_ℓ, γ_{ij} the interface between cells c_i and c_j, and $\Gamma_{k\ell}$ the interface between blocks B_k and B_ℓ. This strategy is particularly effective if the coarse-scale transport solver uses a grid that adapts to flow patterns, i.e., a different grid than what is used by the flow solver, see (Krogstad et al., 2009). How to construct such flow-adapted grids is discussed in detail by Hauge et al. (2010). Here, we show one example:

Example 6 (Flow-Based Coarsening) *We continue the demonstration of the MsMFE method using the PEBI grid from Example 5 to simulate an incompressible two-phase water flooding scenario. More realistic petrophysical parameters have been obtained by sampling porosity and permeability values from the fluvial layers* 50–60 *of the 10th SPE Comparative Solution Project (Christie and Blunt, 2001). The reservoir is initially filled with oil and produced by positioning injection and production wells in an approximate quarter five-spot configuration. The injector is constrained by a total reservoir volume rate of* $500\,\mathrm{m}^3$ *(roughly* 2.8% *of the total model pore volume) per day, while the producer is constrained by a bottom-hole pressure target of* $150\,\mathrm{bar}$*. Figure 10 shows the saturation field obtained at* $2.5\,\mathrm{PVI}$ *(pore volumes injected) using two separate discretisation techniques implemented in MRST. For the fine-scale reference solution in the left plot, we have used the mimetic flow solver in combination with the single-point, implicit, upwind solver* (11)*. For the coarse-scale problem, however, we have combined the MsMFE flow solver with the transport solver* (12) *on a nonuniform, flow-adapted grid (Hauge et al., 2010). Using either discretisation we solve the complete problem using* 10 *pressure steps, within each of which we solve the transport equation using* 15 *sub-steps of equal step size.*

The coarse grid employed in the transport solve for the multiscale solution consists of a total of 248 *blocks, each containing between* 16 *and* 402 *cells from the underlying fine-scale PEBI grid (*23,512 *cells). Because the transport equation is solved on a coarse saturation grid, the multiscale solution is clearly less resolved than the reference solution. However, as demonstrated in earlier work (Aarnes et al., 2007; Krogstad et al., 2009), even this reduced resolution may be sufficient to capture and predict flow response in derived quantities such as well-production profiles. Moreover, the computational cost of the multiscale solution is significantly reduced compared to the reference solution. Specifically, the*

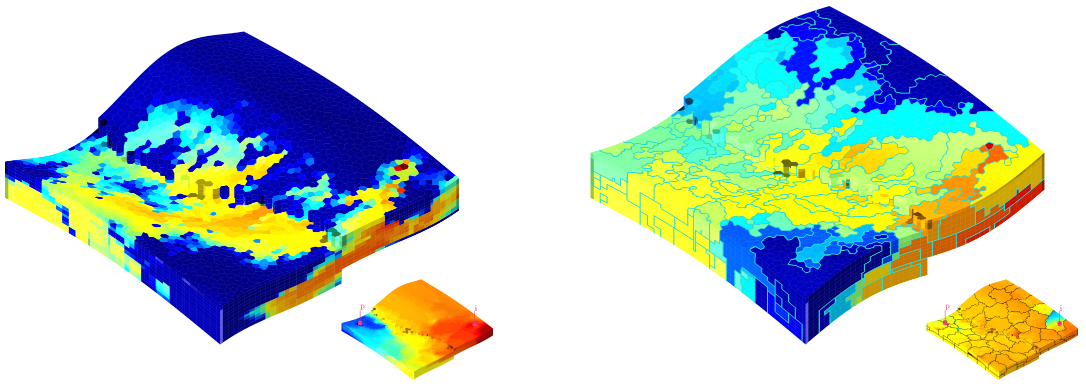

Reference solution MsMFE + transport on flow-adapted grid

Figure 10 *Water flooding scenario realised on a PEBI grid. The left plot shows a fine-scale reference pressure (small plot) and saturation profile, whereas the right plot shows the solution obtained by combining the MsMFE solution (small plot) with the transport solver* (12) *on the flow-adapted grid outlined using enhanced white lines in the big plot.*

total time needed to compute the multiscale solution was 9.0 *and* 8.0 *seconds for the flow and transport solves, respectively. In contrast, the reference solution was computed in* 118.0 *and* 157.0 *seconds for the reference flow and transport solutions. We remark that the reference solution was computed using state-of-the-art linear solvers based on iterative methods preconditioned by an algebraic multigrid-grid technique (Notay, 2010).*

Finally, a recent paper from our research group (Hauge et al., 2010) suggests and analyses techniques for improving the quality of the coarse-scale transport problem. In our experience, much of the added (numerical) diffusion is attributable to bi-directional flow between the blocks in the coarse transport grid. Basing the coarse transport solver on net fluxes rather than fine-scale bi-directional fluxes, reduces the numerical diffusion significantly. However, in Figure 10 we used bi-directional fluxes for illustration purposes.

Faults can either act as conduits for fluid flow in subsurface reservoirs or create flow barriers and introduce compartmentalisation that severely affects fluid distribution and/or reduces recovery. To make flow simulations reliable and predictive, it is crucial to understand and accurately account for the impact that faults have on fluid flow.

Faults affect the transmissibilities geometrically by introducing new cell-to-cell connections and by changing the contact areas between two cells that are juxtaposed over a fault surface. Similarly, the effective transmissibility is changed by fault-zone material interposed between cross-fault juxtaposed cells. To capture these effects, it is common to introduce transmissibility multipliers that account for the reduced or increased permeability for each cross-fault connection. Fault multipliers are typically calculated as empirical functions of the fault displacement, the fault thickness (as a function of displacement), and the clay fraction of the sequence that has moved past each point on the fault, see e.g., (Yielding et al., 1997; Manzocchi et al., 1999; Sperrevik et al., 2002). These multipliers are highly grid dependent and strictly associated with a connection between two grid cells rather than with the fault itself. Consequently, fault multipliers are not a good solution from a modelling point of view because any given multiplier value will be tied to a particular discretisation and introducing these values into another consistent scheme will produce large point-wise errors.

In MRST we assign a unique pressure value to each fault face and explicitly impose flux continuity across the fault. This process leads to a modified discrete system of the form

$$\begin{bmatrix} \tilde{\boldsymbol{B}} & \boldsymbol{C} & \boldsymbol{D} \\ \boldsymbol{C}^{\mathsf{T}} & 0 & 0 \\ \boldsymbol{D}^{\mathsf{T}} & 0 & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{u} \\ -\boldsymbol{p} \\ \boldsymbol{\pi} \end{bmatrix} = \begin{bmatrix} 0 \\ \boldsymbol{q} \\ 0 \end{bmatrix} \tag{13}$$

where the modified inner product matrix $\tilde{\boldsymbol{B}}$ is the ordinary inner product matrix, $\boldsymbol{B}$, of equation (5) modified by an additive contribution of a diagonal matrix incorporating fault transmissibility effects. Additional expository detail concerning this framework, and a related model based on representing faults and shale layers as sealing or semi-permeable internal boundaries are given in a recent paper by Nilsen et al. (2010).

Example 7 (Effects of Flow Barriers in a Real-Field Model) *We now demonstrate the effects of fault and shale layer barriers to flow on a real-field model from the Norwegian Sea. The barriers are represented in the model description as non-negative transmissibility multipliers less than unity. In Figure 11, we show pressure field and corresponding streamlines resulting from a two-phase incompressible flow calculation in a 2D cartoon model derived from the full 3D model, with faults from the original modelled as internal boundaries, depicted in black. The flow is driven by a set of injection and production wells shown in red. The left plots show the results obtained when faults do not impose restrictions to flow, i.e., when all transmissibility multipliers equal unity. We can see that the pressure field is smooth and the streamlines pass through the faults. In the right plots, the faults are completely sealing, producing a discontinuous pressure field and streamline paths that are aligned with the faults.*

Figure 12 shows the pressure field of the original 3D model. Again, the flow is driven from a set of injection and production wells, not shown in the figure. The left plot shows the result obtained with unit transmissibility multipliers for the faults and shale layers, while the right-hand plot shows the pressure resulting from imposing the transmissibility multipliers from the original model description. We clearly observe that the sealing, or partially sealing, faults and shales induce a distinct discontinuity in the pressure field.

The public version of MRST currently supports single-phase and two-phase flow. Our in-house version of MRST has compressible black-oil fluid models implemented, along with several flow and transport solvers. In the next example, we will demonstrate the use of one of our experimental multiscale flow solvers to simulate compressible flow on a geological model with industry-standard complexity.

Example 8 (Primary Production from a Gas Reservoir) *In this example, we will briefly discuss the use of the MsMFE method to compute primary production from a gas reservoir. As our geological model, we will use one of the shallow-marine realisations from the SAIGUP study (Manzocchi et al., 2008). In our current in-house implementation of compressible black-oil models in the MsMFE method, compressibility is handled using a mixed residual formulation*

$$\begin{bmatrix} \boldsymbol{B} & \boldsymbol{C} \\ \boldsymbol{C}^{\mathsf{T}} & \boldsymbol{P} \end{bmatrix} \begin{bmatrix} \boldsymbol{u}_{\mathrm{ms}} + \hat{\boldsymbol{u}}^{\nu+1} \\ \boldsymbol{p}_{\mathrm{ms}} + \hat{\boldsymbol{p}}^{\nu+1} \end{bmatrix} = \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{P}\boldsymbol{p}^{n} + \boldsymbol{V}\boldsymbol{u}^{\nu} \end{bmatrix},$$

for which the elliptic multiscale basis functions act as predictors and a parabolic correction is computed using a standard (non)overlapping Schwarz domain-decomposition method. The method then iterates on the multiscale solution and the corrections until the fine-scale residual is below a prescribed tolerance. The geological model consists of $40 \times 120 \times 20$ *cells (see Figure 13) and is assumed to be initially filled with an ideal gas at* $200\,\mathrm{bar}$ *pressure. The reservoir is produced by a single producer, operating at a constant bottom-hole pressure of* $150\,\mathrm{bar}$. *Figure 13 compares the fine-scale reference solution with multiscale solutions for different tolerances; as our measure, we have used the rate in the well perforation. Except for the coarsest tolerance,* $5 \cdot 10^{-2}$, *the multiscale solution appears to overlap with the fine-scale reference solution.*

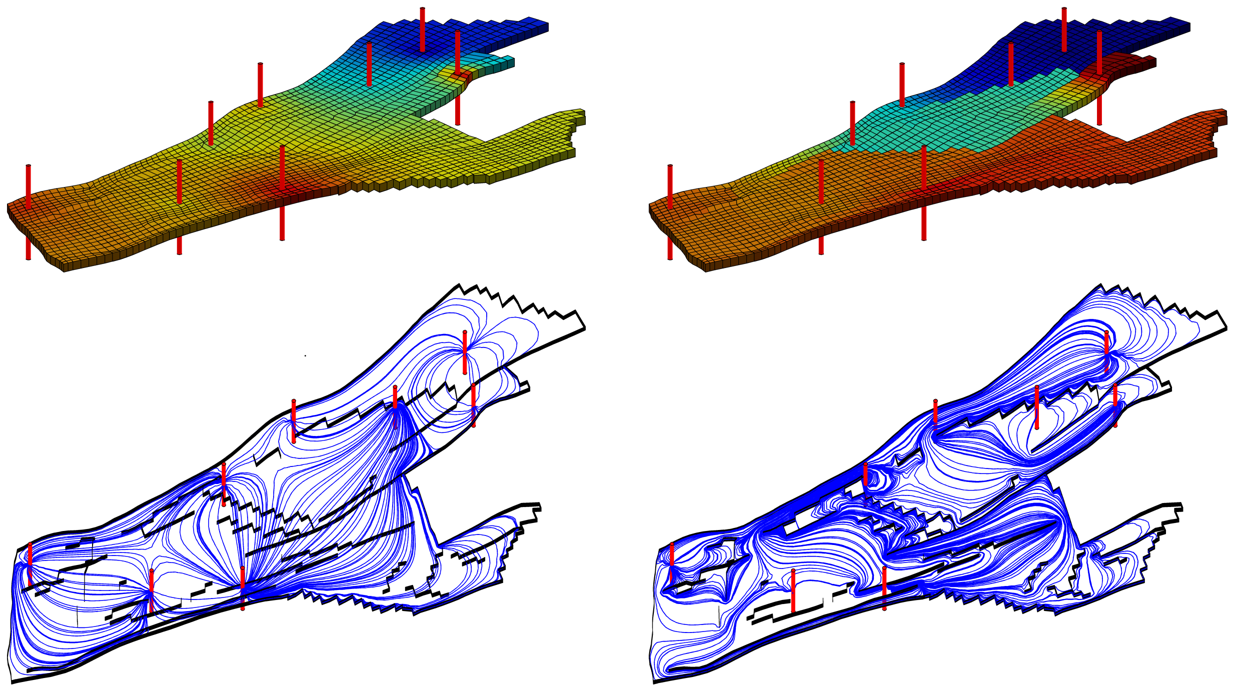


Figure 11 *Two-dimensional mock-up of a real-field model from the Norwegian Sea. The transmissibility multipliers can be used to adjust the impact of a fault from completely transparent (left) to fully sealing (right).*

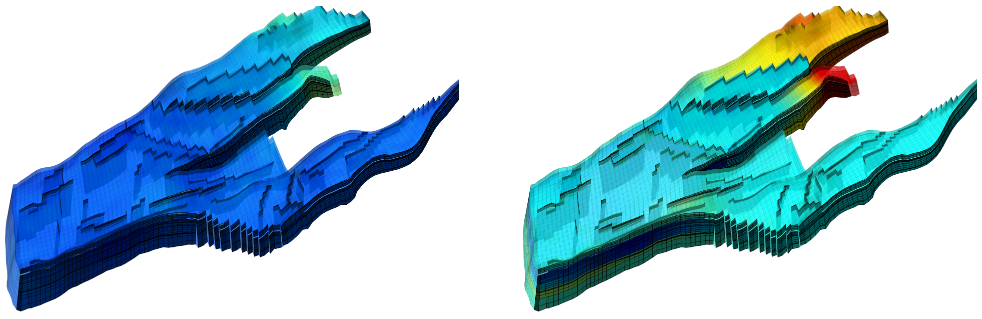


Figure 12 *Effect of flow barriers in a real-field model from the Norwegian Sea—both transparent faults and shales (left), and partially sealing faults and shales (right).*

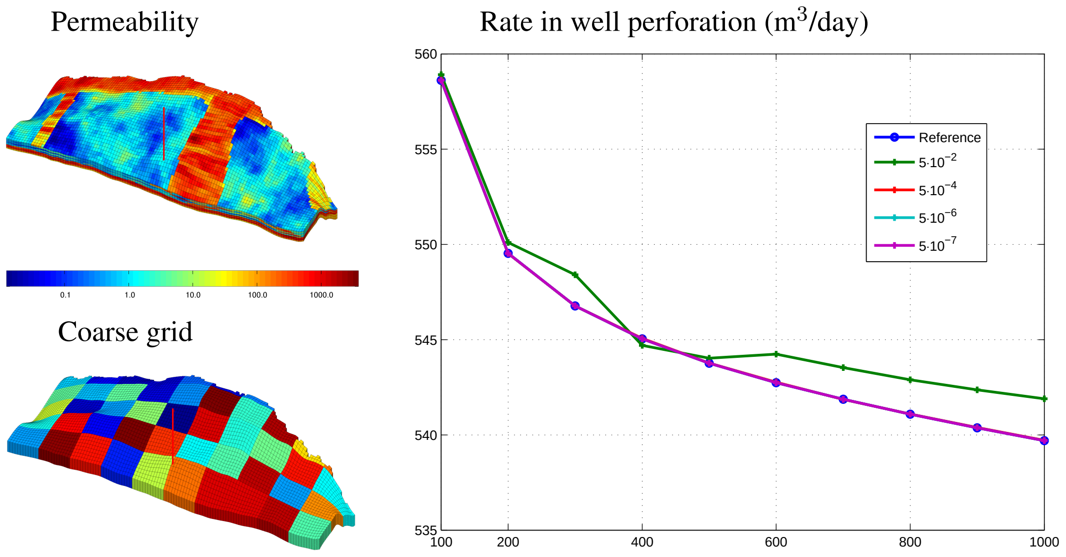


Figure 13 *Primary production from a gas reservoir computed by the MsMFE method.*

Many reservoir management challenges can be cast as mathematical optimisation problems. Examples include data assimilation where a misfit function is to be minimised, and finding optimal well controls to maximise some economic performance measure. A popular approach to obtaining objective function gradients is the use of adjoint equations. The adjoint system of equations is the transpose of the linearised forward system, and accordingly has similar complexity as the forward simulation. Our in-house version of MRST includes an adjoint code for a sequential forward simulation using the mimetic discretization for the pressure equation and an implicit version of the scheme (10) for saturation. In addition, an adjoint code for the multiscale method combined with the flow based coarsening approach is included, see (Krogstad et al., 2009).

Example 9 (Optimising Net Present Value (NPV)) *In this example we consider a synthetic model with two horizontal injectors and one multilateral producer, see Figure 14. We attempt to maximise a simple NPV function; the oil revenue is $100 per barrel and our costs are realised through water injection and production, each $10 per barrel. This means that when the water cut in the producer exceeds* ≈ 0.82*, we are no longer making money. We compare three production strategies:*

1. *The producer operates at constant BHP until the water cut reaches* 0.82*, and the well is shut.*
2. *We assume that each producer well-segment is equipped with an inflow control device (ICD). The producer operates at constant BHP and whenever the water cut in a segment exceeds* 0.82*, the corresponding ICD shuts. The process is continued until all ICDs are shut.*
3. *We use the adjoint code to find optimal segment rates corresponding to a local maximum of the NPV-function.*

The initial simulation input is constant and equal rate for the two injectors and constant BHP for the producer. The initial simulation time is set to 500 *days which is equivalent to* $1.25\,\mathrm{PVI}$*. In Figure 14 we show the accumulated NPV for the three production scenarios. We observe that for production scenario 1, the well is shut down after about* 225 *days with an achieved NPV of \$* 72.8 *million. Scenario 2 is equal to scenario 1 until the first ICD is shut, and the improvement obtained by being able to shut down individual segments of the well is evident. All ICDs are shut after about 375 days, and the obtained NPV is \$* 79.9 *million. Finally, in scenario 3, water cut is not taken into account, only maximising the NPV. The obtained NPV is \$* 92.6 *million.*

Outlook

Replicablility is a fundamental scientific principle that is currently receiving increasing attention among computational scientists. The level of sophistication and complexity in new computational methods makes it difficult for researchers to implement methods published by others and replicate their results.

We believe that releasing the MATLAB Reservoir Simulation Toolbox under the GNU General Public License (GPL) is an important step for our group towards supporting the idea of reproducible science, thereby allowing others to more easily benefit from our efforts. Although we are not yet there, we are aiming towards a state where our publications on new computational methodologies can be accompanied by open-access software that can be used by others to verify our computational results.

We note in conclusion that many of the examples presented in the section on “Advanced Examples” employ features of MRST which currently exist only in our in-house, development version of the package. However, we do expect to release several of these capabilities in the next public edition of MRST, currently scheduled for late September or early October 2010.

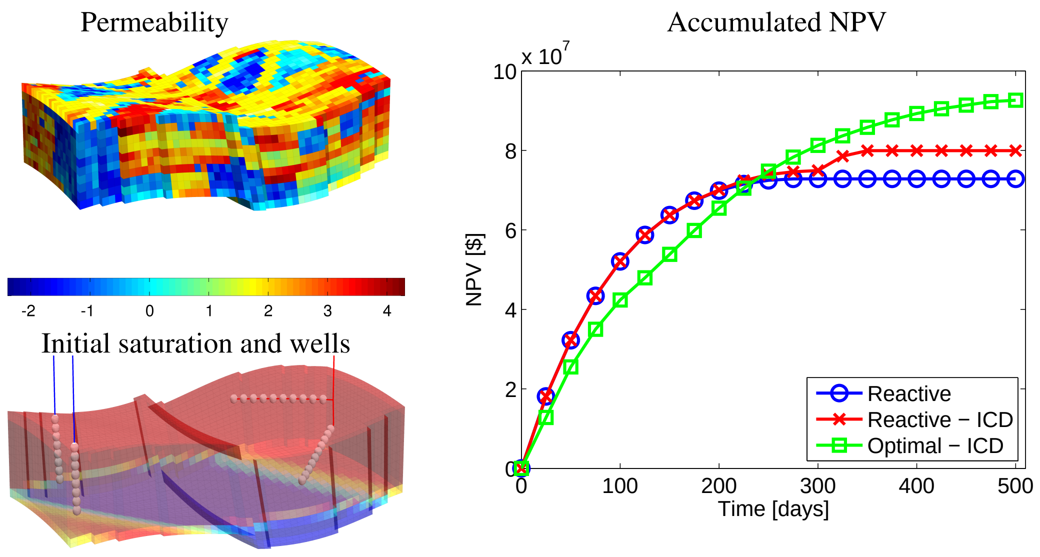


Figure 14 *Accumulated net present value for the three production strategies.*

Acknowledgements

The research was funded in part by the Research Council of Norway through grants no. 158908, 174551, 175962, 178013, and 186935.

References

Aarnes, J.E., Hauge, V.L. and Efendiev, Y. [2007] Coarsening of three-dimensional structured and unstructured grids for subsurface flow. *Adv. Water Resour.*, **30**(11), 2177–2193, doi:10.1016/j.advwatres.2007.04.007.

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

Aarnes, J.E., Krogstad, S. and Lie, K.A. [2006] A hierarchical multiscale method for two-phase flow based upon mixed finite elements and nonuniform coarse grids. *Multiscale Model. Simul.*, **5**(2), 337–363 (electronic), ISSN 1540-3459.

Aarnes, J.E., Krogstad, S. and Lie, K.A. [2008] Multiscale mixed/mimetic methods on corner-point grids. *Comput. Geosci.*, **12**(3), 297–315, ISSN 1420-0597, doi:10.1007/s10596-007-9072-8.

Aavatsmark, I. [2002] An introduction to multipoint flux approximations for quadrilateral grids. *Comput. Geosci.*, **6**, 405–432, doi:10.1023/A:1021291114475.

Aavatsmark, I., Barkve, T., Bøe, Ø. and Mannseth, T. [1994] Discretization on non-orthogonal, curvilinear grids for multi-phase flow. *Proc. of the 4th European Conf. on the Mathematics of Oil Recovery*.

Aziz, K. and Settari, A. [1979] *Petroleum Reservoir Simulation*. Elsevier Applied Science Publishers, London and New York.

Brezzi, F., Lipnikov, K. and Simoncini, V. [2005] A family of mimetic finite difference methods on polygonial and polyhedral meshes. *Math. Models Methods Appl. Sci.*, **15**, 1533–1553, doi:10.1142/S0218202505000832.

Brezzi, F. and Fortin, M. [1991] *Mixed and Hybrid Finite Element Methods*, vol. 15 of *Springer Series in Computational Mathematics*. Springer-Verlag, New York, ISBN 0-387-97582-9.

Chen, Z. and Hou, T. [2003] A mixed multiscale finite element method for elliptic problems with oscillating coefficients. *Math. Comp.*, **72**, 541–576.

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

Deutsch, C.V. and Journel, A.G. [1998] *GSLIB: Geostatistical software library and user's guide*. Oxford University Press, New York, 2nd edn.

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

Edwards, M.G. and Rogers, C.F. [1994] A flux continuous scheme for the full tensor pressure equation. *Proc. of the 4th European Conf. on the Mathematics of Oil Recovery*.

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

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

Hauge, V.L., Lie, K.A. and Natvig, J.R. [2010] Flow-based grid coarsening for transport simulations. *Proceedings of ECMOR XII–12th European Conference on the Mathematics of Oil Recovery*, EAGE, Oxford, UK.

Hou, T. and Wu, X.H. [1997] A multiscale finite element method for elliptic problems in composite materials and

porous media. *J. Comput. Phys.*, **134**, 169–189.
Jenny, P., Lee, S.H. and Tchelepi, H.A. [2003] Multi-scale finite-volume method for elliptic problems in subsurface flow simulation. *J. Comput. Phys.*, **187**, 47–67.
Karypis, G. and Kumar, V. [1995] A fast and high quality multilevel scheme for partitioning irregular graphs. *International Conference on Parallel Processing*, 113–122.
Kippe, V., Aarnes, J.E. and Lie, K.A. [2008] A comparison of multiscale methods for elliptic problems in porous media flow. *Comput. Geosci.*, **12**(3), 377–398, ISSN 1420-0597, doi:10.1007/s10596-007-9074-6.
Krogstad, S., Hauge, V.L. and Gulbransen, A.F. [2009] Adjoint multiscale mixed finite elements. *SPE Reservoir Simulation Symposium, The Woodlands, TX, USA, 2–4 February 2009*, doi:10.2118/119112-MS.
Lipnikov, K., Shashkov, M. and Yotov, I. [2009] Local flux mimetic finite difference methods. *Numer. Math.*, **112**(1), 115–152, doi:10.1007/s00211-008-0203-5.
Manzocchi, T., Walsh, J.J., Nell, P. and Yielding, G. [1999] Fault transmissiliblity multipliers for flow simulation models. *Petrol. Geosci.*, **5**, 53–63.
Manzocchi, T. et al. [2008] Sensitivity of the impact of geological uncertainty on production from faulted and unfaulted shallow-marine oil reservoirs: objectives and methods. *Petrol. Geosci.*, **14**(1), 3–15.
Natvig, J.R. and Lie, K.A. [2008a] Fast computation of multiphase flow in porous media by implicit discontinuous Galerkin schemes with optimal ordering of elements. *J. Comput. Phys.*, **227**(24), 10108–10124, ISSN 0021-9991, doi:10.1016/j.jcp.2008.08.024.
Natvig, J.R. and Lie, K.A. [2008b] On efficient implicit upwind schemes. *Proceedings of ECMOR XI, Bergen, Norway, 8–11 September*, EAGE.
Natvig, J.R. et al. [2009] Multiscale mimetic solvers for efficient streamline simulation of fractured reservoirs. *SPE Reservoir Simulation Symposium, The Woodlands, TX, USA, 2–4 February 2009*, doi:10.2118/119132-MS.
Nilsen, H., Lie, K.A., Natvig, J. and Krogstad, S. [2010] Accurate modelling of faults by multipoint, mimetic, and mixed methods, submitted to the SPE Journal.
Notay, Y. [2010] An aggregation-based algebraic multigrid method. *Electronic Transactions on Numerical Analysis*, **37**, 123–146.
Skaflestad, B. and Krogstad, S. [2008] Multiscale/mimetic pressure solvers with near-well grid adaption. *Proceedings of ECMOR XI–11th European Conference on the Mathematics of Oil Recovery*, A36, EAGE, Bergen, Norway.
Sperrevik, S., Gillespie, P.A., Fisher, Q.J., Halverson, T. and Knipe, R.J. [2002] Empirical estimation of fault rock properties. In: Koestler, A.G. and Hunsdale, R. (Eds.) *Hydrocarbon seal quantification: papers presented at the Norwegian Petroleum Society Conference, 16-18 October 2000, Stavanger, Norway*. Elsvier, vol. 11 of *Norsk Petroleumsforening Special Publication*, 109–125.
Yielding, G., Freeman, B. and Needham, D.T. [1997] Quantitative fault seal prediction. *AAPG Bull*, **81**(6), 897–917.
Zhou, H. and Tchelepi, H. [2008] Operator-based multiscale method for compressible flow. *SPE J.*, **13**(2), 267–273.