Open Source MATLAB Implementation of Consistent Discretisations on Complex Grids

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

Received: date / Accepted: date

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. It is well known that conventional two-point flux-approximation methods are only consistent for K orthogonal grids and will therefore not converge in the general case. In recent years, there has been significant research into consistent and convergent methods, including mixed, multipoint, and mimetic discretisation methods. Likewise, so-called multiscale methods based upon hierarchically coarsened grids have received a lot of attention.

The paper does not propose novel mathematical methods but instead presents an open-source MATLAB® toolkit that can be used as an efficient test platform for (new) discretisation and solution methods in reservoir simulation. The aim of the toolkit is to support reproducible research and simplify the development, verification and validation, and testing and comparison of new discretisation and solution methods on general unstructured grids, including in particular corner-point and 2.5D PEBI grids. The toolkit consists of a set of data structures and routines for creating, manipulating, and visualising petrophysical data, fluid models, and (unstructured) grids, including support for industry-standard input formats, as well as routines for computing single and multiphase (incompressible) flow. We review key features of the toolkit and discuss a generic mimetic formulation that includes many known discretisation methods, including both the standard two-point method as well as consistent and convergent multipoint and mimetic methods. Apart from the core routines and data structures, the toolkit contains add-on modules that implement more advanced solvers and functionality. Herein, we show examples of multiscale methods and adjoint methods for use in optimisation of rates and placement of wells.

1 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 convergent only if each grid cell $\mathbf{K}$-orthogonal. For hexahedral grids, this means that each cell is a parallelepiped and $\vec{n}_{ij}\mathbf{K}_i\vec{n}_{ik} = 0$ in all grid cells i (here, $\mathbf{K}$ is the permeability tensor in cell i and $\vec{n}_{ij}$ and $\vec{n}_{ik}$ denote normal vectors into two neighbouring cells). Ensuring K-orthogonality is difficult when representing particular geological features like sloping faults, horizontal wells, etc. Hence, there has in recent years been significant research into mixed [8], multipoint [5], and mimetic [9] 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 as-

tonishing 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 [16, 13]. 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 [17, 15, 18] 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 our 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. Secondly, we present a class of mimetic methods that incorporates several well-known discretisation methods as special cases on simple grids while at the same time providing consistent discretisation on grids that are not K-orthogonal. Finally, 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, and applications such as optimal control or well placement.

2 The MATLAB Reservoir Simulation Toolbox

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

Grids: a common data structure and interface for all types of grids (unstructured representation); no grid generator but grid factory routines for rectilinear grids, triangular and tetrahedral grids, 2D Voronoi grids, extrusion of areal grids to 2.5D volumetric grids, etc; tutorial examples and a few realistic data sets

Input and output: routines for reading and processing industry-standard input files for grids, petrophysical parameters, fluid models, wells, boundary conditions, simulation setup, etc.

Parameters: a data structure for petrophysical parameters (and a few, very simplified geostatistical routines); common interface to fluid models (Version 2011a supports incompressible fluid models, but in-house development version also has support for compressible black-oil fluids, which will likely be released in future versions of MRST); routines for setting and manipulating boundary conditions, sources/sinks, well models, etc

Units: MRST works in strict SI units but supports conversion to/from other unit systems like field-units, etc. Unless reading from an industry-standard input format, the user is responsible for explicit conversion and consistency of units[1].

Reservoir state: data structure for pressure, fluxes, saturations, ...

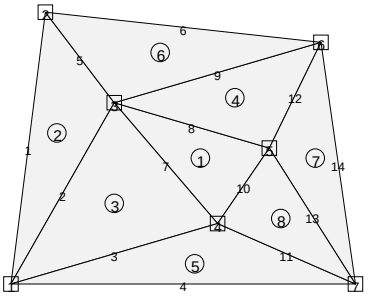
Postprocessing: visualisation routines for scalar cell and face data, etc

Solvers: 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 [4], working on unstructured, polyhedral grids

Linear algebra: MRST relies on MATLAB's builtin linear solvers but these can easily be replaced by specialised solvers using standard MATLAB conventions for doing so.

We will now go into more details about some of the components outlined above. All code excerpts and explicit statements of syntax refer to release 2011a of MRST. Complete scripts and the data necessary to run most of the examples in the paper can be downloaded from the MRST webpage [26]. The interested reader should also review the tutorials included in the current release.

[1] All examples considered herein are computed using SI units, but other units may be used when reporting parameters and solutions.



cells.faces =

1	10
1	8
1	7
2	1
2	2
2	5
3	3
3	7
3	2
4	8
4	12
4	9
5	3
5	4
5	11
6	9
6	6
6	5
7	13
7	14
:	:

faces.nodes =

1	1
1	2
2	1
2	3
3	1
3	4
4	1
4	7
5	2
5	3
6	2
6	6
7	3
7	4
8	3
8	5
9	3
9	6
10	4
10	5
:	:

faces.neighbors =

0	2
2	3
3	5
5	0
6	2
0	6
1	3
4	1
6	4
1	8
8	5
4	7
7	8
0	7

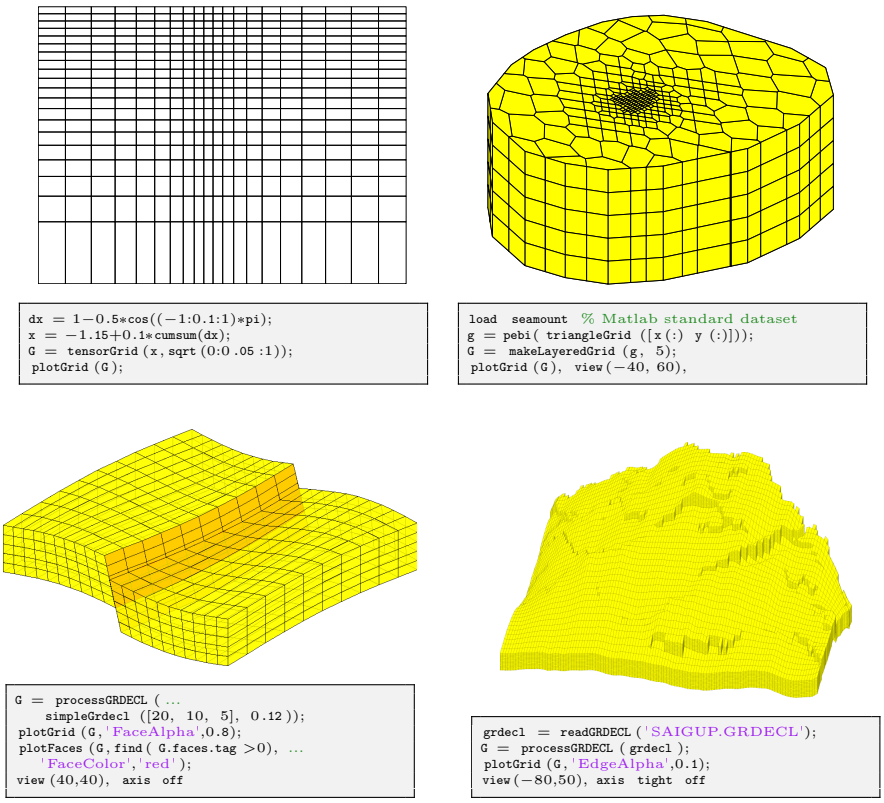
Figure 1: Illustration of the `cell` and `faces` fields of the grid structure: cell numbers are marked by circles, node numbers by squares, and face numbers have no marker. A zero value in `face.neighbors` means that the faces lies at an outer boundary and hence only has one neighbour.

2.1 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 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 vertices 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 `cells.indexMap` that maps internal cell indexes to external cell indexes, which is useful e.g., if the model contains inactive cells[2]. Likewise, the `faces` structure contains the

[2]Inactive cells are specified in industry-standard input format using special keywords (e.g., ACTNUM in the widely used ECLIPSE standard). Alternatively, inactive cells in `clist` can



```
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 % Matlab standard dataset
g = pebi( triangleGrid ([x(:) y(:)]));
G = makeLayeredGrid(g, 5);
plotGrid(G), view(-40, 60),
```

```
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('SAIGUP.GRDECL');
G = processGRDECL( grdecl );
plotGrid(G,'EdgeAlpha',0.1);
view(-80,50), axis tight off
```

Figure 2: Examples of grids and the MRST statements necessary to create them. The upper-left plot shows a standard tensor-product Cartesian grid. In the upper-right plot we 2.5D PEBI grid extruded from the unstructured point set in the data file `seamount.mat` that is distributed with MATLAB. The lower-left plot shows an example of a corner-point grid with a single sloping fault and wavy top- and bottom surfaces. The lower-right plot shows one of the grid models supplied together with MRST.

number of global faces, an array `faces.nodes` of vertices connected to each face, 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

be explicitly removed through a call to `removeCells(G,clist)`.

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 four examples of grids and the commands necessary to create and display them. The rectilinear grid is generated by the grid-factory routine `tensorGrid`, which takes one vector of grid nodes per spatial dimension as input and creates the corresponding tensor-product mesh. To generate the 2.5D PEBI grid, we start with an unstructured point set (`seamount.mat`) and use `triangleGrid` to generate a Delauny grid, from which a 2D Voronoi grid is generated using `pebi`. Then this areal grid is extruded to a 3D model consisting of five layers using the routine `makeLayeredGrid`. In the third plot, we have used the routine `simpleGrdecl` to generate an example of an ECLIPSE input stream and then `processGRDECL` to process the input stream and generate a corner-point grid. The lower-right plot shows a realistic reservoir model where we have used `readGRDECL` to read the grid section of an ECLIPSE input deck; the routine reads the file and creates an input stream on the same format as `simpleGrdecl`.

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)`.

2.2 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. Whereas petrophysical parameters are often supplied for all cells (active and inactive) in a model, the `rock` and grid structures in MRST only represent the active cells. The porosity field `rock.poro` is therefore a vector with one value for each active cell in the corresponding grid model. For models with inactive cells, the field `cells.indexMap` in the grid structure contains the indices of the active cells sorted in ascending order. If `p` contains porosity values for all cells, this porosity distribution is assigned to the `rock` structure by the call `rock.poro = p(G.cells.indexMap)`.

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 three or six columns for a symmetric, full tensor permeability (in two and three spatial dimensions, respectively). 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

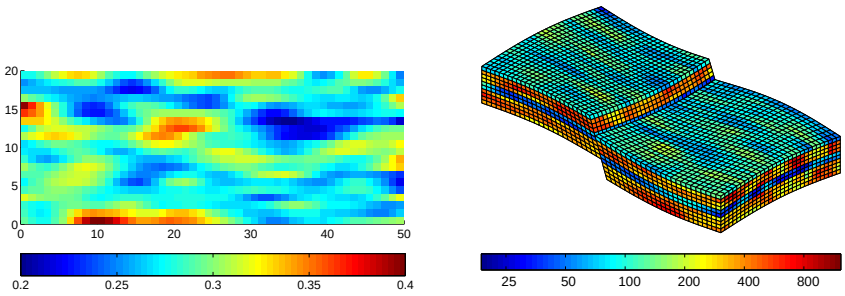


Figure 3: Two examples of MRST's simplified geostatistics. The left plot shows a 50×20 porosity field generated as a Gaussian field with a larger filter size in the x-direction than in the y-direction. The right plot shows a stratigraphic grid with a single fault and four geological layers, each with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

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.

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 start with the Carman–Kozeny relation*

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

and 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 $A_v = 6/d_p$. Assuming further that $\tau = 0.81$ gives us an explicit formula for calculating the isotropic permeability K from the porosity ϕ. 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 layers 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 mD*,* 400 mD*,* 50 mD*, and* 350 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 layers, 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 [12] or commercial software for geological modelling.

2.3 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,

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

Here, $\vec{v}$ denotes the Darcy velocity, p pressure, and $\mathbf{K}$ 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 one-sided 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 [7], the lowest-order mixed finite-element methods [8], multipoint flux approximation schemes [6, 14, 5], and recently developed mimetic finite-difference methods [9]. Two-point discretisations give diagonal transmissibility matrices and are not convergent for general grids. Mixed, multipoint, and mimetic methods are consistent and convergent on non-orthogonal grids, but lead to full matrices $\boldsymbol{T}_i$. Such schemes will be discussed in more detail in Section 3; for now we only assume that there exists a consistent scheme of the form (2) that is convergent for fully unstructured, polyhedral grids.

In the following, we 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 imposed to ease the

presentation and give a uniform formulation of a large class of schemes. The underlying principles may be applied to any reasonable scheme. By augmenting (2) with flux and pressure continuity across cell faces, we obtain the following linear system [8]

$$\begin{bmatrix} \boldsymbol{B} & \boldsymbol{C} & \boldsymbol{D} \\ \boldsymbol{C}^{\mathsf{T}} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{D}^{\mathsf{T}} & \boldsymbol{0} & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{u} \\ -\boldsymbol{p} \\ \boldsymbol{\pi} \end{bmatrix} = \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{q} \\ \boldsymbol{0} \end{bmatrix}. \tag{3}$$

Here, the first row in the block-matrix equation corresponds to Darcy's law in the form (2) for all grid cells, the second row corresponds to mass conservation for all cells, whereas the third row expresses continuity of fluxes for all cell faces. 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 (3) can be solved using a Schur-complement method and MATLAB's standard linear solvers or third-party linear system solver software such as AGMG [29]. A block-wise Gaussian elimination for (3) yields a positive-definite system (the Schur complement) for the face pressures,

$$\big(\boldsymbol{D}^{\mathsf{T}}\boldsymbol{B}^{-1}\boldsymbol{D} - \boldsymbol{F}^{\mathsf{T}}\boldsymbol{L}^{-1}\boldsymbol{F}\big)\boldsymbol{\pi} = \boldsymbol{F}^{\mathsf{T}}\boldsymbol{L}^{-1}\boldsymbol{q}, \tag{4}$$

where $\boldsymbol{F} = \boldsymbol{C}^{\mathsf{T}}\boldsymbol{B}^{-1}\boldsymbol{D}$ and $\boldsymbol{L} = \boldsymbol{C}^{\mathsf{T}}\boldsymbol{B}^{-1}\boldsymbol{C}$. Given the face pressures, the cell pressures and fluxes can be reconstructed by back-substitution, i.e., solving

$$\boldsymbol{L}\boldsymbol{p} = \boldsymbol{q} + \boldsymbol{F}\boldsymbol{\pi}, \qquad \boldsymbol{B}\boldsymbol{u} = \boldsymbol{C}\boldsymbol{p} - \boldsymbol{D}\boldsymbol{\pi}.$$

Here, the matrix $\boldsymbol{L}$ is by construction diagonal and computing fluxes is therefore an inexpensive operation. It is also worth noting that we only need $\boldsymbol{B}^{-1}$ in the solution procedure above. Many schemes—including the mimetic method, the MPFA-O method, and the standard two-point scheme—yield algebraic approximations for the $\boldsymbol{B}^{-1}$ matrix. Thus, (3) encompasses a family of discretisation schemes whose properties are determined by the choice of $\boldsymbol{B}$, which we will discuss in more detail in Section 3.1.

2.4 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 a discretisation as introduced in the previous section to solve the single-phase pressure equation

$$\nabla \cdot \vec{v} = q, \qquad \vec{v} = -\frac{\mathsf{K}}{\mu}\big[\nabla p + \rho g \nabla z\big]. \tag{5}$$

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());
```

Flow solvers in MRST 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 (3), with a mimetic discretisation, 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 boundary conditions, 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.

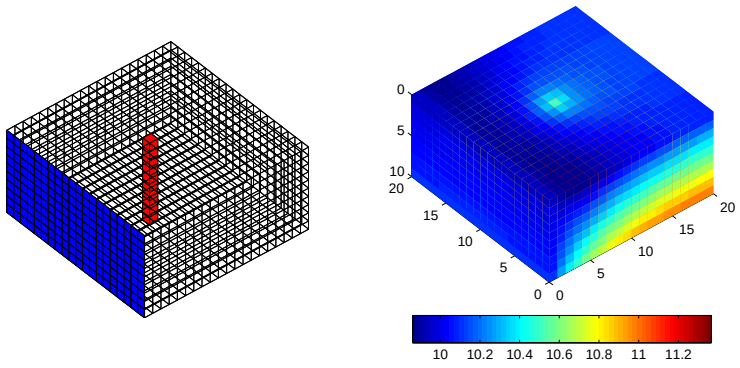


Figure 4: Example of a simple flow driven by a column of source cells and a Dirichlet boundary condition. The left plot shows the model setup and the right plot the corresponding pressure solution.

2.5 Two-Phase Flow

Two-phase incompressible flow of a wetting and non-wetting fluid can be described by the following system of equations [10] (or see the derivation in [1]):

$$\nabla \cdot \vec{v} = q, \qquad \vec{v} = -\mathbf{K}\big[\lambda \nabla p + (\lambda_w \rho_w + \lambda_n \rho_n) g \nabla z\big], \tag{6}$$

$$\phi \frac{\partial s_w}{\partial t} + \nabla \cdot \Big(f_w(s_w)\big[\vec{v} + \lambda_n(\rho_n - \rho_w) g \mathbf{K} \nabla z\big] \Big) = q_w. \tag{7}$$

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 (6)–(7) 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, (6) is solved with fixed saturation values to provide pressure and fluxes, and then the fluxes are used to evolve the saturations according to (7). 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 (6). MRST supports two basic saturation solvers that both use a single-point upwind discretisation (i.e., the single-point upstream-mobility scheme for unidirectional flow). Dropping subscripts to denote phases and assuming no gravity, they can be written in the form

$$s_i^{n+1} = s_i^n + \frac{\Delta t}{\phi_i |c_i|}\Big(\max(q_i, 0) + f(s_i^m)\min(q_i, 0)\Big) - \frac{\Delta t}{\phi_i |c_i|}\Big(\sum_j \big[f(s_i^m)\max(v_{ij}, 0) + f(s_j^m)\min(v_{ij}, 0)\big]\Big), \tag{8}$$

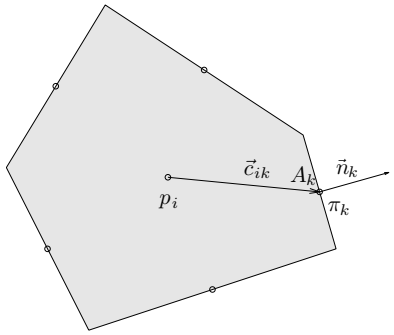


Figure 5: The quantities used to define the mimetic inner product or equivalently the transmissibility in a single polyhedral cell: p_i cell pressure, π_k face pressure, $\vec{n}_k$ normal of face, A_k area of face and $\vec{c}_{ik}$ vector from cell centroid to face centroid.

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 (6), capillary effects can be included by defining the pressure as the *global* pressure, $p = p_n - p_c$, where the so-called complementary pressure [10] 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 2011a. In the next section, we will go into more details about consistent and convergent discretisations on unstructured polyhedral grids, before we end the paper with an overview of how the resulting solvers can be applied to more advanced examples, including solvers and functionality that are not yet released publicly.

3 Mimetic Discretisation Methods

In this section we will discuss the mimetic method in more detail. We start by discussing the inner product, which can be used to design the properties of the method. Then we give a short discussion of Peaceman-type well models for the method.

The mimetic method (see [9]) is defined in terms of a local inner product $\boldsymbol{M}$ or equivalently an inverse inner product (or transmissibility matrix) $\boldsymbol{T}$, which gives the cell-based discretisation of Darcy's law

$$\boldsymbol{M}\boldsymbol{u} = \boldsymbol{e}p_i - \boldsymbol{\pi}, \quad \vec{e} = (1, \ldots, 1)^{\mathsf{T}}, \qquad \boldsymbol{u} = \boldsymbol{T}(\boldsymbol{e}p_i - \boldsymbol{\pi}). \tag{9}$$

Here $\boldsymbol{\pi}$ is the pressure at the face centroids and p the cell pressure or equivalently

the pressure at the cell centroid as illustrated in Figure 5. Mimetic methods are constructed so that they are exact for linear pressure fields and give a symmetric positive-definite matrix $\boldsymbol{M}$. In addition, the methods use discrete pressure and fluxes associated with cell and face centroids, respectively, and consequently resemble finite-difference methods.

A linear pressure field can be written in the form $p = \vec{x} \cdot \vec{a} + b$ for a constant vector $\vec{a}$ and scalar b, giving a Darcy velocity equal $\vec{v} = -\mathbf{K}\vec{a}$. Let $\vec{n}_k$ denote the area-weighted normal vector to face number k and $\vec{c}_{ik}$ be the vector pointing from the cell centroid to the corresponding face centroid, as seen in Figure 5. Then the flux and pressure drop are given by

$$u_k = -\vec{n}_k \mathbf{K} \vec{a}, \qquad p_i - \pi_j = \vec{c}_{ik} \cdot \vec{a}. \tag{10}$$

Inserting this into (9), we see that the matrices $\boldsymbol{M}$ and $\boldsymbol{T}$ must satisfy the following consistency conditions

$$\boldsymbol{MNK} = \boldsymbol{C}, \qquad \boldsymbol{NK} = \boldsymbol{TC}, \tag{11}$$

where each row $\boldsymbol{c}_i^{\mathsf{T}}$ of the matrix $\boldsymbol{C}$ corresponds to $\vec{c}_{ik}$, and each row $\boldsymbol{n}_i^{\mathsf{T}}$ of $\boldsymbol{N}$ corresponds to $\vec{n}_k$, see [9] for the discrete flux case.

From the local discretisation on each cell, the global stiffness matrix $\boldsymbol{B}$ in (3) is assembled as a block-diagonal matrix in which each block is the inner product of the corresponding cell. For two-phase flow, the inner product is multiplied by the inverse total mobility for each cell. As we have seen, the system (3) can be reduced to a linear system that only involves the face pressures $\boldsymbol{\pi}$, using a transformation that requires the computation of $\boldsymbol{B}^{-1}$, which is a block matrix that can be assembled from the inverse inner product T. In the following, we describe a few inner products (and inverse inner products) while emphasising aspects of implementation and discuss specific properties of the different discretisations. The implementation can be found in `computeMimeticIP.m` in MRST [26].

3.1 Inner Products

In the original method [9], inner products of discrete velocities are considered. In reservoir simulation however, it is more common to consider the face fluxes as unknowns. Accordingly, we will henceforth consider inner products of fluxes rather than velocities. We note here that the relation between the two is trivial, as an inner product of velocities becomes an inner product for fluxes by pre- and post-multiplying by the inverse area of the corresponding faces. Let $\boldsymbol{A}$ be the diagonal matrix with a_{ii} the face area of the i-th face. Then the flux inner product $\boldsymbol{M}_{\text{flux}}$ is related to the velocity inner product $\boldsymbol{M}_{\text{vel}}$ through

$$\boldsymbol{M}_{\text{flux}} = \boldsymbol{A}^{-1} \boldsymbol{M}_{\text{vel}} \boldsymbol{A}^{-1}. \tag{12}$$

Henceforth we will only consider inner products for fluxes.

To yield a consistent discretisation of (1), an inner product matrix $\boldsymbol{M}$ or an inverse inner product $\boldsymbol{T}$ must result in a discretisation that is exact for linear

pressures, i.e., fulfils (11). To derive a family of valid solutions, we first observe the following key geometrical property (see [9])

$$\boldsymbol{C}^{\mathsf{T}}\boldsymbol{N} = \mathrm{diag}(|\Omega_i|), \tag{13}$$

which relates $\boldsymbol{C}$ and $\boldsymbol{N}$ as follows on general polyhedral cells. Multiplying the first equation of (11) by $\boldsymbol{K}^{-1}\boldsymbol{C}^{\mathsf{T}}\boldsymbol{N}$, we derive the relation

$$\boldsymbol{M}\boldsymbol{N} = \frac{1}{|\Omega_i|}\boldsymbol{C}\boldsymbol{K}^{-1}\boldsymbol{C}^{\mathsf{T}}\boldsymbol{N},$$

from which it follows that the family of valid solutions has the form

$$\boldsymbol{M} = \frac{1}{|\Omega_i|}\boldsymbol{C}\boldsymbol{K}\boldsymbol{C}^{\mathsf{T}} + \boldsymbol{M}_2, \tag{14}$$

in which $\boldsymbol{M}_2$ is a matrix defined such that $\boldsymbol{M}$ is symmetric positive definite and $\boldsymbol{M}_2\boldsymbol{N} = \boldsymbol{0}$. Any symmetric and positive definite inner product fulfilling these requirements can be represented in the compact form

$$\boldsymbol{M} = \frac{1}{|\Omega_i|}\boldsymbol{C}\boldsymbol{K}^{-1}\boldsymbol{C}^{\mathsf{T}} + \boldsymbol{Q}_N^{\perp}\boldsymbol{S}_M{\boldsymbol{Q}_N^{\perp}}^{\mathsf{T}}, \tag{15}$$

where $\boldsymbol{Q}_N^{\perp}$ is an orthonormal basis for the null space of $\boldsymbol{N}^{\mathsf{T}}$, and $\boldsymbol{S}_M$ is any symmetric, positive-definite matrix. In the code and the following examples, however, we will instead use a null space projection $\boldsymbol{P}_N^{\perp} = \boldsymbol{I} - \boldsymbol{Q}_N{\boldsymbol{Q}_N}^{\mathsf{T}}$, where $\boldsymbol{Q}_N$ is a basis for the space spanned by the columns of $\boldsymbol{N}$. Similarly, the inverse inner products take the form

$$\boldsymbol{T} = \frac{1}{|\Omega_i|}\boldsymbol{N}\boldsymbol{K}\boldsymbol{N}^{\mathsf{T}} + \boldsymbol{Q}_C^{\perp}\boldsymbol{S}{\boldsymbol{Q}_C^{\perp}}^{\mathsf{T}}, \tag{16}$$

where $\boldsymbol{Q}_C^{\perp}$ is an orthonormal basis for the nullspace of $\boldsymbol{C}^{\mathsf{T}}$ and $\boldsymbol{P}_C^{\perp} = \boldsymbol{I} - \boldsymbol{Q}_C{\boldsymbol{Q}_C}^{\mathsf{T}}$ is the corresponding nullspace projection.

The matrices $\boldsymbol{M}$ and $\boldsymbol{T}$ in (15) and (16) are evidently symmetric so only an argument for the positive definiteness is in order. Writing $\boldsymbol{M}$ in (15) as $\boldsymbol{M} = \boldsymbol{M}_1 + \boldsymbol{M}_2$, positive semi-definiteness of $\boldsymbol{M}_1$ and $\boldsymbol{M}_2$ imply that $\boldsymbol{M}$ is positive semi-definite. Let $\boldsymbol{z}$ be an arbitrary non-zero vector which we split (uniquely) as $\boldsymbol{z} = \boldsymbol{N}\tilde{\boldsymbol{z}} + \boldsymbol{z}'$ where $\boldsymbol{N}^{\mathsf{T}}\boldsymbol{z}' = \boldsymbol{0}$. If $\boldsymbol{z}' = \boldsymbol{0}$, we have $\boldsymbol{z}^{\mathsf{T}}\boldsymbol{M}\boldsymbol{z} = \tilde{\boldsymbol{z}}^{\mathsf{T}}\boldsymbol{N}^{\mathsf{T}}\boldsymbol{M}_1\boldsymbol{N}\tilde{\boldsymbol{z}} > 0$ since $\boldsymbol{C}^{\mathsf{T}}\boldsymbol{N}$ has full rank. If $\boldsymbol{z}' \neq \boldsymbol{0}$, we have $\boldsymbol{z}^{\mathsf{T}}\boldsymbol{M}\boldsymbol{z} = \boldsymbol{z}^{\mathsf{T}}\boldsymbol{M}_1\boldsymbol{z} + {\boldsymbol{z}'}^{\mathsf{T}}\boldsymbol{M}_2\boldsymbol{z}' > 0$ since $\boldsymbol{z}^{\mathsf{T}}\boldsymbol{M}_1\boldsymbol{z} \geq 0$ and ${\boldsymbol{z}'}^{\mathsf{T}}\boldsymbol{M}_2\boldsymbol{z}' > 0$. An analogous argument holds for the matrix $\boldsymbol{T}$.

We will now outline a few specific and valid choices that are implemented in the basic flow solvers of MRST. We start by considering the one-dimensional case with the cell $[-1, 1]$ for which $\boldsymbol{N} = \boldsymbol{C} = [1, -1]^{\mathsf{T}}$ and $\boldsymbol{Q}_N^{\perp} = \boldsymbol{Q}_C^{\perp} = \frac{1}{\sqrt{2}}[1, 1]^{\mathsf{T}}$. Hence

$$\begin{aligned}
\boldsymbol{M} &= \frac{1}{2}\begin{bmatrix} 1 \\ -1 \end{bmatrix}\frac{1}{K}[1, -1] + \frac{1}{2}\begin{bmatrix} 1 \\ 1 \end{bmatrix}S_M[1, 1], \\
\boldsymbol{T} &= \frac{1}{2}\begin{bmatrix} 1 \\ -1 \end{bmatrix}K\,[1, -1] + \frac{1}{2}\begin{bmatrix} 1 \\ 1 \end{bmatrix}S\,[1, 1].
\end{aligned} \tag{17}$$

The structure of the inner product should be invariant under scaling of K and thus we can write the inner product as a one-parameter family of the form

$$\begin{aligned} \boldsymbol{M} &= \frac{1}{2K}\left(\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} + \frac{2}{t}\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}\right), \\ \boldsymbol{T} &= \frac{K}{2}\left(\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} + \frac{t}{2}\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}\right). \end{aligned} \tag{18}$$

In the following, we will use this one-dimensional expression together with transformation properties of the inner product to look at the correspondence between mimetic, TPFA, and Raviart–Thomas methods.

Two-point type methods. The TPFA method is the gold standard for practical reservoir simulation despite its theoretical shortcomings. Since the TPFA discretisation requires a diagonal tensor, it is easily seen from (11) that this is only possible in the case when the vectors $\boldsymbol{K}\vec{n}_i$ and $\vec{c}_i$ are parallel, i.e., the grid is K-orthogonal. In any case (K-orthogonality or not), we define the diagonal TPFA transmissibility tensor $\boldsymbol{T}$ by

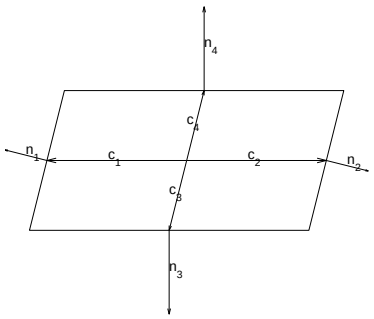
$$\boldsymbol{T}_{ii} = \vec{n}_i \cdot \boldsymbol{K}\vec{c}_i/|\vec{c}_i|^2. \tag{19}$$

This defines the unique TPFA method for K-orthogonal grids[3]. The extension to non-orthogonal grids is not unique, and the TPFA method does not give a consistent discretisation in this case. Because of its simplicity, the TPFA method is strictly monotone if $T_{ii} > 0$, which implies that the fluxes form a directed acyclic graph, a property that can be used to accelerate the solution of the transport equations considerably, as discussed in [28].

When written in its standard form, the TPFA method is cell-centred and thus less computationally costly than face-centred methods that arise from a consistent hybrid mimetic formulation. However, since the method is not consistent, it will in many cases be necessary to investigate the grid-orientation effects of the TPFA method before using it for realistic simulations. We therefore present a mimetic-type discretisation that coincides with the TPFA method in its region of validity, while at the same time giving a valid mimetic inner product for all types of grids and permeability tensors. This minimises the need for investigating other effects of the TPFA method, such as errors introduced by corners and well modelling. An advantage of this method compared to using, e.g., an MPFA method is that the implementation is simpler for general unstructured grids. We refer to the corresponding method as IP_QTPFA. To derive the method, we consider a cuboid cell and insert (10) into (9) defined for a single face to obtain

$$T_{ii}\vec{c}_i \cdot \vec{a} = \vec{n}_i \mathsf{K} \vec{a}.$$

[3]As written, this does not always yield a positive value for the two-point transmissibility. To ensure positive transmissibilities it is normal to define the face centroids as the arithmetic mean of the associated corner-point nodes and the cell centroids as the arithmetic mean of the top and bottom surface centroids. For most reservoir grids this gives a positive transmissibility.



$$\vec{c}_1 + \vec{c}_2 = 0, \quad \vec{c}_3 + \vec{c}_4 = 0$$

$$\Longrightarrow \boldsymbol{Q}_C^{\perp} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{bmatrix}$$

$$\vec{n}_1 + \vec{n}_2 = 0, \quad \vec{n}_3 + \vec{n}_4 = 0$$

Figure 6: Creation of nullspace of $\boldsymbol{C}$ for a parallelepiped. The left plot shows the vectors $\vec{c}_i$ and $\vec{n}_i$ (the latter scaled by a factor 0.3).

Next, we set $\vec{a} = \vec{n}_i$ and use the property that $\vec{c}_i \cdot \vec{n} = \frac{1}{2}|\Omega_i|$ for cuboid cells to derive an expression for T_{ii}, which can be written in matrix form and equated with (14)

$$\boldsymbol{T}_{\text{TPF}} = \frac{2}{|\Omega_i|} \operatorname{diag}(\boldsymbol{N}\boldsymbol{K}\boldsymbol{N}^{\mathsf{T}}) = \frac{1}{|\Omega_i|} \boldsymbol{N}\boldsymbol{K}\boldsymbol{N}^{\mathsf{T}} + \boldsymbol{T}_2. \tag{20}$$

Here, $\boldsymbol{T}_2$ should be invariant under $\boldsymbol{P}_C^{\perp}$. Moreover, one can easily show that $\boldsymbol{P}_C^{\perp}\boldsymbol{N} = \boldsymbol{0}$ for parallelepipeds (see Figure 6), which means that we can define the generalised TPFA inner product as

$$\begin{aligned} \boldsymbol{T} &= \frac{1}{|\Omega_i|}\Big[\boldsymbol{N}\boldsymbol{K}\boldsymbol{N}^{\mathsf{T}} + 2\,\boldsymbol{P}_C^{\perp} \operatorname{diag}(\boldsymbol{N}\boldsymbol{K}\boldsymbol{N}^{\mathsf{T}})\boldsymbol{P}_C^{\perp}\Big], \\ \boldsymbol{M} &= \boldsymbol{T}^{-1}. \end{aligned} \tag{21}$$

Equation (1) is invariant when space and permeability are transformed as

$$\vec{x} \mapsto \boldsymbol{S}\vec{x} \quad \text{and} \quad \mathsf{K} \mapsto \boldsymbol{S}^{\mathsf{T}}\mathsf{K}\boldsymbol{S}, \tag{22}$$

respectively. We emphasise that the inner product (21) evidently is invariant under the transformations (22), but will only be diagonal for K-orthogonal grids. In the one-dimensional case, (21) simplifies to

$$\boldsymbol{T} = \begin{bmatrix} K & 0 \\ 0 & K \end{bmatrix}, \tag{23}$$

which is the TPFA expression for this case. The same result is obtained by setting $t = 2$ in the one-dimensional expression (18). Since the mimetic and the TPFA method do not couple dimensions for orthogonal grids with diagonal tensor, this derivation is also valid for rectangular cuboids.

Raviart–Thomas (RT0). The following inner product is equivalent to the mixed Raviart–Thomas inner product for grids that are orthogonal and has the same principal axes as the permeability tensor

$$\boldsymbol{M} = \frac{1}{|\Omega_i|}\boldsymbol{C}\boldsymbol{K}\boldsymbol{C}^{\mathsf{T}} + \frac{|\Omega_i|}{6}\boldsymbol{P}_N^{\perp}\big[\operatorname{diag}(\boldsymbol{N}\boldsymbol{K}\boldsymbol{N}^{\mathsf{T}})\big]^{-1}\boldsymbol{P}_N^{\perp}. \tag{24}$$

This can be verified by a direct calculation for the one-dimensional case, which reduces to

$$M = \frac{1}{K}\begin{bmatrix} 2/3 & -1/3 \\ -1/3 & 2/3 \end{bmatrix}, \tag{25}$$

which coincides with (18) for $t = 6$. The fact that this inner product does not couple different directions for orthogonal grids and diagonal tensors implies that (24) is also equal to the lowest-order Raviart–Thomas (RT0) method on (rectangular) cuboids. We refer to this inner product as 'IP_QRT'. The corresponding quasi-inverse, which is the exact inverse for orthogonal grids, reads

$$\boldsymbol{T} = \frac{1}{|\Omega_i|}\Big[\boldsymbol{N}\boldsymbol{K}\boldsymbol{N}^{\mathsf{T}} + 6\,\boldsymbol{P}_C^{\perp}\,\mathrm{diag}(\boldsymbol{N}\boldsymbol{K}\boldsymbol{N}^{\mathsf{T}})\boldsymbol{P}_C^{\perp}\Big]. \tag{26}$$

This inner product will also, by the transformation property, be equal to the Raviart–Thomas formulation for all cases that can be transformed to the above case by an affine transformation of the form (22). Using a mimetic inner product which is simpler to calculate and equal to the mixed RT0 inner product for all grid cells is not possible because of the need to integrate the nonlinear function introduced by the determinant of the Jacobian of the mapping from the grid cell to the unit cell where the RT0 basis functions are defined.

Local-flux mimetic MPFA. In addition to the above methods, which are all based on the assumption of a positive-definite inner product and exactness for linear flow, the MPFA method can be formulated as a mimetic method. This was first done by [21, 24] and is called the local-flux mimetic formulation of the MPFA method. In this case, all faces of a cell are such that each corner is associated with d unique faces, where d is the dimension. The inner product of the local-flux mimetic method gives exact result for linear flow and is block diagonal with respect to the faces corresponding to each corner of the cell, but it is not symmetric. The block-diagonal property makes it possible to reduce the system into a cell-centred discretisation for the cell pressures. This naturally leads to a method for calculating the MPFA transmissibilities. The crucial point is to have the corner geometry in the grid structure and handle the problems with corners which do not have three unique half faces associated. Currently the MPFA-O method is formulated in such a way [21] and work has been done for the MPFA-L method, but in this case the inner product will vary over the different corners. Our implementation of MPFA is based upon a combination of MATLAB and C and is therefore available as an add-on module to MRST.

Parametric family. We notice that the (inverse) inner products IP_QTPFA in (21) and IP_QRT in (26) differ only by a constant in front of the regularisation part (the second term). Both methods belong to a family whose inverse inner product can be written in the form

$$\begin{aligned} \boldsymbol{T} &= \frac{1}{|\Omega_i|}\Big[\boldsymbol{N}\boldsymbol{K}\boldsymbol{N}^{\mathsf{T}} + t\,\boldsymbol{P}_C^{\perp}\,\mathrm{diag}(\boldsymbol{N}\boldsymbol{K}\boldsymbol{N}^{\mathsf{T}})\boldsymbol{P}_C^{\perp}\Big] \\ \boldsymbol{M} &= \boldsymbol{T}^{-1}, \end{aligned} \tag{27}$$

in which t is a parameter that can be varied continuously from zero to infinity. In MRST, this family of inner products is called 'IP_QFAMILY' and the parameter t is supplied in a separate option

```
S = computeMimeticIP(G, rock, 'Verbose', true,...
                     'InnerProduct', 'ip_qfamily', 'qparam', t);
```

IP_SIMPLE. For historical reasons, the default inner product used in MRST reads

$$\begin{aligned} \boldsymbol{Q} &= \operatorname{orth}(\boldsymbol{A}^{-1}\boldsymbol{N}) \\ \boldsymbol{M} &= \frac{1}{|\Omega_i|}\boldsymbol{C}\boldsymbol{K}^{-1}\boldsymbol{C}^{\mathsf{T}} + \frac{d|\Omega_i|}{6\operatorname{tr}(\boldsymbol{K})}\boldsymbol{A}^{-1}(\boldsymbol{I}-\boldsymbol{Q}\boldsymbol{Q}^{\mathsf{T}})\boldsymbol{A}^{-1} \end{aligned} \tag{28}$$

with the approximate inverse

$$\begin{aligned} \boldsymbol{Q} &= \operatorname{orth}(\boldsymbol{A}\boldsymbol{C}) \\ \boldsymbol{T} &= \frac{1}{|\Omega_i|}\Big[\boldsymbol{N}\boldsymbol{K}\boldsymbol{N}^{\mathsf{T}} + \frac{6}{d}\operatorname{tr}(\boldsymbol{K})\boldsymbol{A}(\boldsymbol{I}-\boldsymbol{Q}\boldsymbol{Q}^{\mathsf{T}})\boldsymbol{A}\Big]. \end{aligned} \tag{29}$$

This inner product was used in [4] inspired by [9] and was chosen to resemble the mixed Raviart–Thomas inner product (they are equal for scalar permeability on orthogonal grids, which can be verified by inspection. Since this inner product is based on velocities, it involves pre- and post-multiplication of (inverse) face areas, and it might not be obvious that it fits into the formulations (15)–(16). However, after a small computation one is convinced that the second part of (15) is invariant under multiplication by $\boldsymbol{P}_N^{\perp}$ so its eigenspace corresponding to the nonzero eigenvalues must be equal to the nullspace of $\boldsymbol{N}^{\mathsf{T}}$. A similar argument holds for the inverse inner product.

Example 2 (Grid-Orientation Effects and Monotonicity) *It is well known that two-point approximations are convergent only in the special case of K-orthogonal grids. Mimetic and multipoint schemes, on the other hand, are constructed to be consistent and convergent for rough grids and full-tensor permeabilities, but may suffer from pressure oscillations for full-tensor permeabilities and large anisotropy ratios and/or high aspect ratio grids. In this example, we use one of the example grids supplied with MRST to illustrate these two observations. The routine* `twister` *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 non-orthogonal, but smoothly varying, logically Cartesian grid.*

To investigate grid-orientation effects, we impose a permeability tensor with anisotropy ratio $1:1000$ *aligned with the* x*-axis and compute the flow resulting from a prescribed horizontal pressure drop and no-flow conditions on the top and bottom boundaries. Figure 7 shows pressure computed on a* 100×100 *grid by*

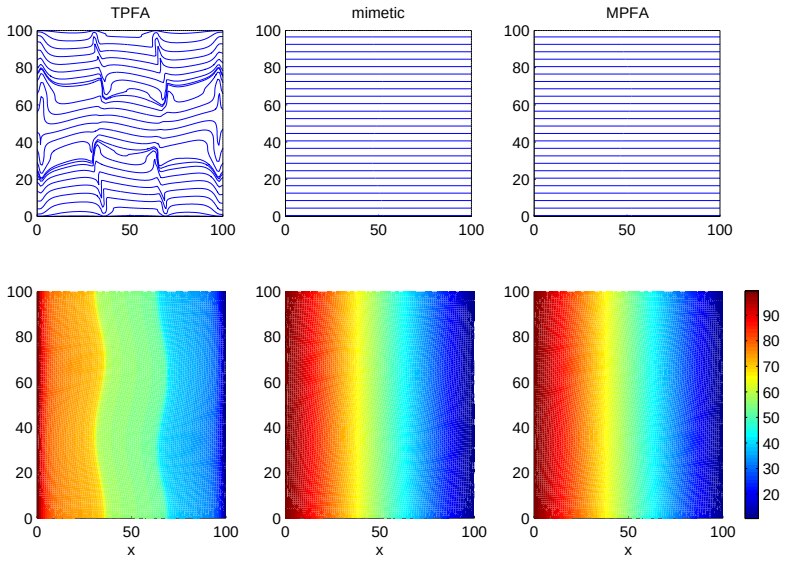


Figure 7: Grid-orientation effects for a homogeneous domain with Dirichlet boundary conditions (left,right) and no-flow conditions (top, bottom) computed with three different pressure solvers in MRST. The permeability field is homogeneous with anisotropy ratio $1:1000$ aligned with the grid. The upper row shows streamlines visualising the velocity field, whereas the bottom row shows the pressure field.

the TPFA method, the mimetic method with inner product IP_SIMPLE, and the local-flux mimetic version of the MPFA-O method. To visualise the corresponding velocity field, we also show streamlines traced by MRST's implementation of Pollock's method. Whereas the mimetic 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.

In Figure 8, 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 by the TPFA scheme is wrong. In this case, the pressures computed with the mimetic and the MPFA-O schemes appear to be correct and have no discernible oscillations. However, the streamline distributions clearly show that the resulting velocity fields are highly oscillatory, in particular for the mimetic method with the IP_SIMPLE inner product.*

To compare the effect of using an approximate inverse in the inner-product matrices, we have computed the velocity field for the mimetic methods resulting from inner products (29)*,* (24)*, and* (21)*. From the streamline plots shown in Figure 9, it is evident that both IP_SIMPLE and IP_QRT yield more oscillatory velocity fields than the IP_QTPF method.*

Example 3 (Analytical solution) *In this example, we use a classical analytical solution based on complex analysis to verify the numerical solutions obtained*

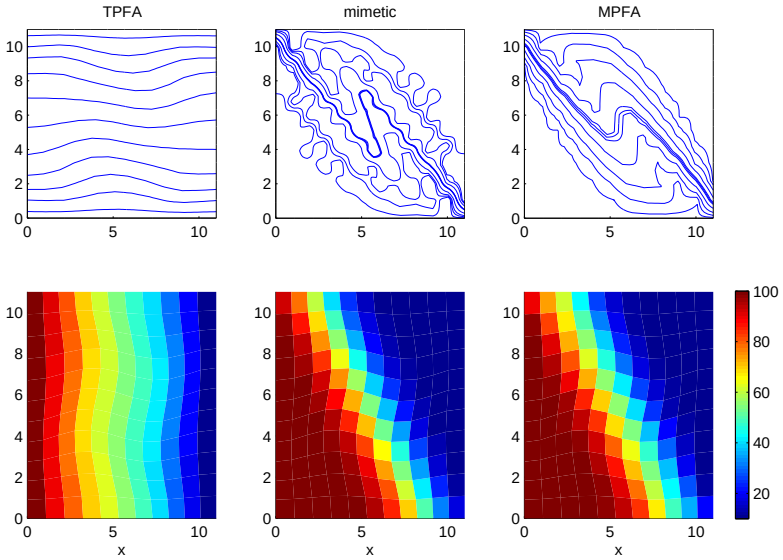


Figure 8: Monotonicity effects demonstrated with the same setup as in Figure 7, but with the anisotropy ratio $1:1000$ making an angle $\pi/6$ with the grid directions.

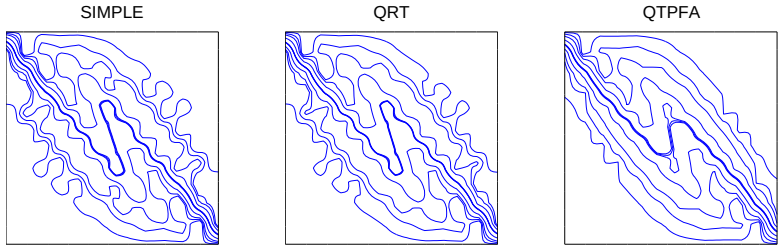


Figure 9: Lack of monotonicity visualised by tracing streamlines for the velocity fields computed by three mimetic pressure solvers in MRST for the same setup as in Figure 8.

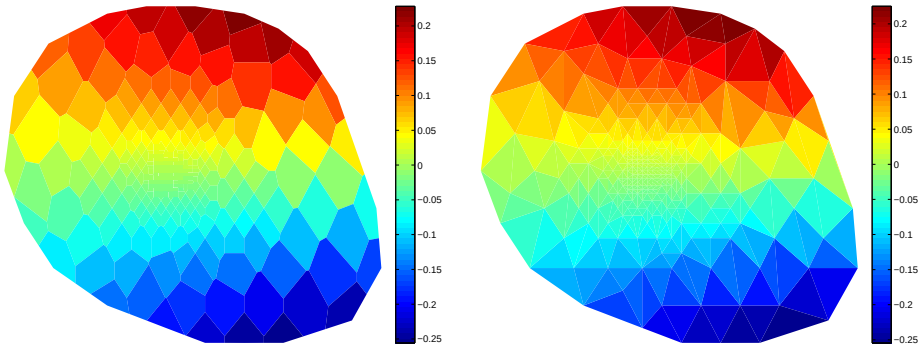
by MRST. A standard method to construct analytical solutions to the Laplacian equation in 2D is to write the unknown as the real or imaginary part of an complex analytic function $f(z)$. We choose the function $f(z) = (z + \frac{1}{2}z^2)$ and set our analytical solution to be $p_a(x, y) = \mathcal{I}f(x + iy) = y + xy$. As our computational domain, we choose a 2D triangular grid generated from the `seamount.mat` *data set (see e.g., the upper-left plot in Figure 2) and its dual Voronoi grid, both centred at the origin. By prescribing $p_a(x, y)$ along the boundary, we obtain a Dirichlet problem with known solution.*

To compute numerical approximations to $p_a(x, y)$ we use the MPFA-O scheme and the mimetic method with inner product 'IP_SIMPLE'. Figure 10 shows the approximate solutions and the discrepancy from the analytical solution at the cell centres for both grids. For the triangular grid, the two methods produce the exact same result because all mimetic methods with a symmetric inner product are identical for cell and face pressures on triangular grids. On the Voronoi grid, the mimetic method is slightly more accurate than the MPFA-O method and both methods have lower errors than on the triangular grid.

Example 4 (Real-Field Model) *In this example, we consider two synthetic models derived from the 3D simulation model of a reservoir from offshore Norway. The original model is given as a $46 \times 112 \times 22$ logically Cartesian corner-point grid with 44,915 active cells. To assess grid-orientation effects introduced by different inner products, we consider two single layers (layer one and ten from the top of the model), which we flatten and modify so that the thickness is constant and all pillars in the corner-point description are vertical. Well positions are assigned by keeping one perforation for each of the original wells. Let (x_i, y_i), $i = 1, \ldots, n$ denote the resulting well positions and q_i the well rates. If we model each well as a logarithmic singularity along a vertical line at (x_i, y_i), the corresponding analytical solution to* (1) *in an infinite domain with constant permeability K is given by*

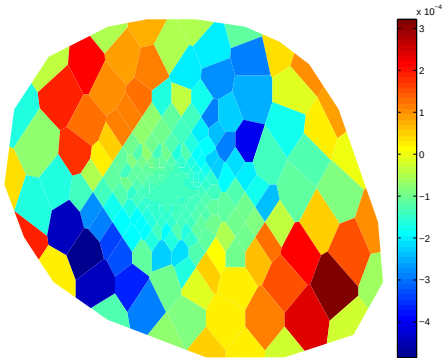
$$p(x, y, z) = \sum_{i=1}^{n} \frac{q_i}{2\pi K} \ln\Big(\sqrt{(x - x_i)^2 + (y - y_i)^2}\Big). \tag{30}$$

To generate a representative analytical solution, we set $K = 500$ mD and prescribe (30) *along the outer lateral boundary of each model. In Figure 11, we compare this analytical solution with numerical solution computed by the TPFA method and the mimetic IP_QRT method for the first layer. Whereas the mimetic solution shows good correspondence with the analytical solution, the TPFA solution exhibits large grid-orientation effects, which are particularly evident between wells one and four and in the region below well two. With the consistent mimetic scheme, the error is localised around the wells, whereas it is up to 10 % and distributed all over the reservoir for the TPFA method. (The full 3D model has a much rougher geometry that contains pinch-outs, sloping faults, and eroded layers, which all will contributed to larger grid-orientation effects for inconsistent methods). In a real simulation, a good well model e.g., as discussed in the next section) can correct for errors in the vicinity of the well, but not global errors as in the inconsistent TPFA method.*

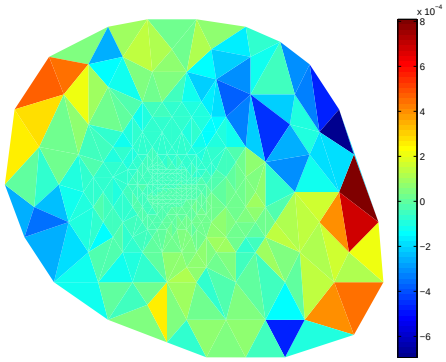


$p_{\text{mim}}(x, y)$ on PEBI grid

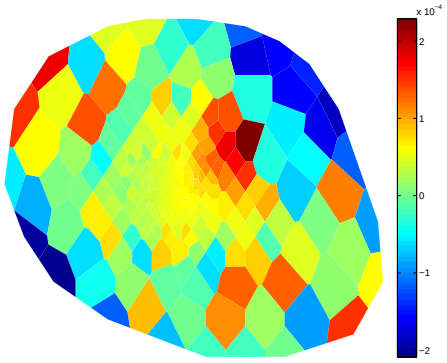
$p_{\text{mim}}(x, y)$ on triangular grid



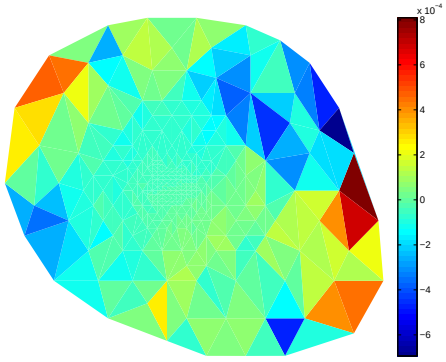
$p_{\text{mim}}(x, y) - p_a(x, y)$



$p_{\text{mim}}(x, y) - p_a(x, y)$



$p_{\text{mpfa}}(x, y) - p_a(x, y)$



$p_{\text{mpfa}}(x, y) - p_a(x, y)$

Figure 10: The upper row shows the pressure for a Dirichlet problem with analytical solution $p_a(x, y) = y + xy$ computed by a mimetic method on a PEBI grid and on a triangular grid. The second and third rows show the error in solutions computed by the mimetic and MPFA-O method, respectively.

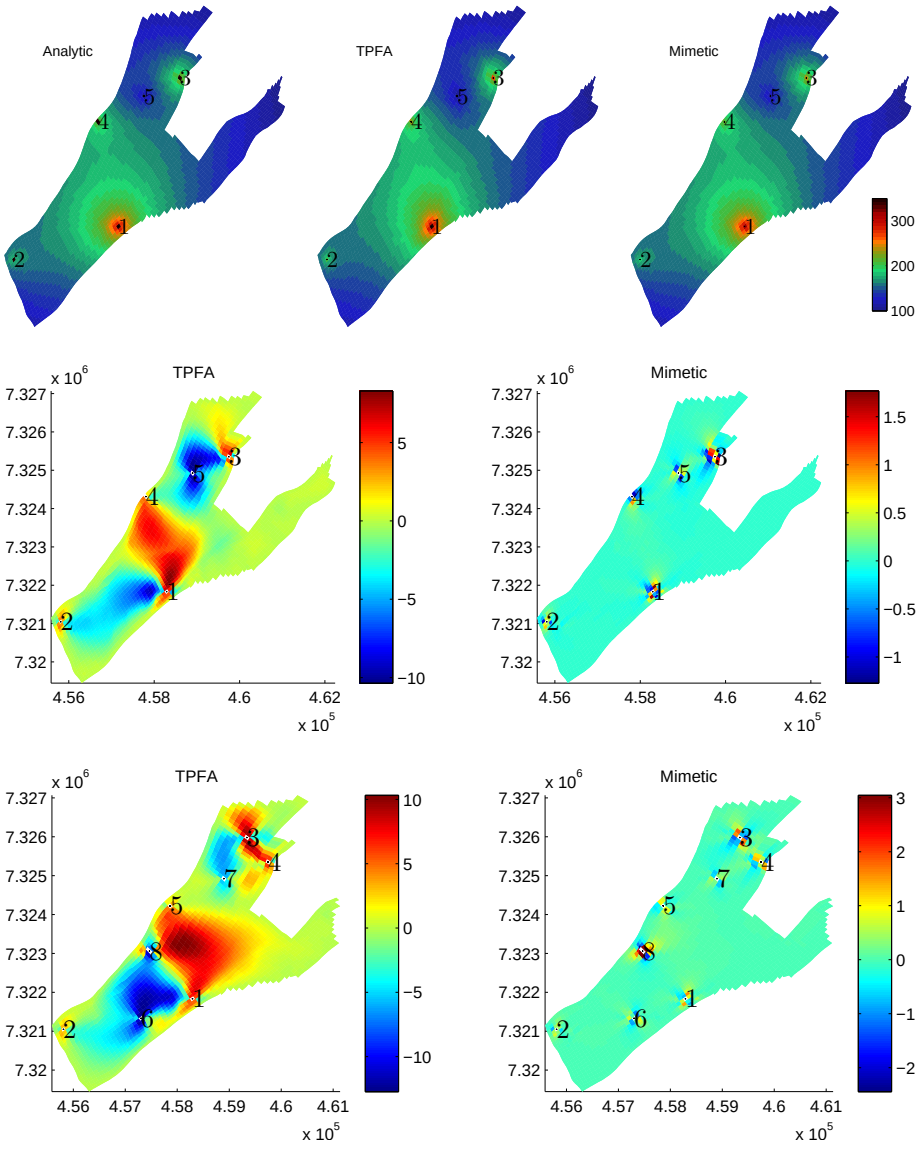


Figure 11: Analytical and numerical solutions for the flattened top layer of a real-field model. The upper row compares pressure solutions (plotted in units 'bar') for layer one, whereas the two lower rows show the cell-wise discrepancies between the analytical and the numerical solutions for layers one and ten.

Table 1: Error as a function of inner product in (27) for the real-field models. The error is defined as the maximum discrepancy in cell pressure scaled by the total span in analytic pressure values outside the well cells. The corresponding errors for the IP_QRT, MPFA-O, and TPFA methods are 0.0096, 0.0158, and 0.0560 for layer one and 0.0127, 0.0123 and 0.053 for layer ten.

Inner product (t)	0.5	1	2	3	4
Layer one	0.0752	0.0394	0.0161	0.0097	0.0080
Layer ten	0.0695	0.0326	0.0119	0.0095	0.0104
Inner product (t)	5	6	7	10	100
Layer one	0.0078	0.0096	0.0118	0.0112	0.0179
Layer ten	0.0114	0.0127	0.0137	0.0152	0.0189

In Table 1 we report the maximum relative error in pressure for various members of the inner-product family introduced above applied to the grid extracted from the tenth layer. For layer ten, the minimum error occurs around $t = 3$*, which is a method somewhere between the quasi-two-point and the quasi-Raviart–Thomas methods. For both grids, the minimal error is smaller than for both the IP_QRT and the MPFA-O method.*

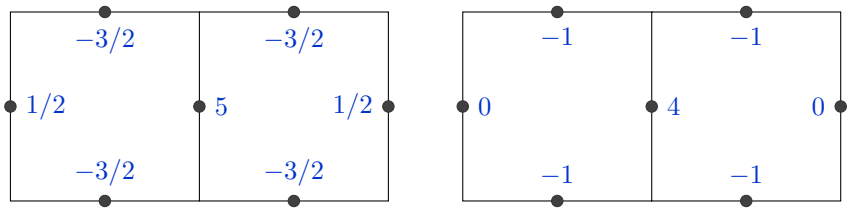
Using the full model, we have also verified (or validated) our implementations against the leading commercial simulator for incompressible, two-phase flow.

3.2 Finite-Difference Stencils

To complete the discussion of the previously presented mimetic finite-difference methods we demonstrate a few of the resulting finite-difference stencils that relate the interface pressure values in (4). In particular, we discretise the pressure equation (1) with permeability $\mathbf{K} = I$ on a two-dimensional, equidistant grid, $\Delta x = \Delta y = 1$ using inner products already implemented in MRST. Figure 12 shows the results. We notice in particular that the 'IP_SIMPLE' inner product of equation (29) and the 'IP_QFAMILY' inner product of equation (27) with parameter $t = 6$ produce stencils that have both positive and negative off-diagonal coupling terms. Moreover, setting $t = 4$ produces a stencil that resembles the classical five-point discretisation scheme of the Laplace operator on a rotated grid. Finally, for $t \in (2, 4)$, all off-diagonal coupling terms are negative. The case of $t = 2$ is identical to the inner product given by 'IP_ TPFA' and 'IP_ QTPFA'. For general grids and permeability, the inner product given by (27) only includes 'IP_ QRT' and 'IP_ QTPFA' since these inner products are the only ones that are invariant under the affine transformations in (22).

3.3 Well Modelling

Wells in reservoirs typically have small diameters compared to the size of the simulation cells, and it is therefore common to employ a well index (productivity index) WI to relate the local flow rate q to the difference between the well



(a) IP_SIMPLE, IP_QFAMILY ($t = 6$)

(b) IP_QFAMILY ($t = 4$)

(c) IP_QFAMILY ($t = 3$)

(d) IP_TPF, IP_QFAMILY ($t = 2$)

Figure 12: Finite-difference stencils for interface pressures when $\Delta x = \Delta y = 1$ and $\mathbf{K} = I$.

pressure p_w and numerically computed pressure p_E in the perforated grid-cell as follows

$$-q = \lambda_t(s_E)\mathrm{WI}(p_E - p_w). \tag{31}$$

Commonly used is Peaceman's well index [30], which for a vertical well in a Cartesian cell with dimensions $\Delta x \times \Delta y \times \Delta z$ is given as

$$\mathrm{WI} = \frac{2\pi k \Delta z}{\ln(r_0/r_w)}. \tag{32}$$

For isotropic media, k is given by $\mathbf{K} = k\mathbf{I}$, and

$$r_0 = 0.14(\Delta x^2 + \Delta y^2)^{\frac{1}{2}}, \tag{33}$$

where r_0 is the *effective well-cell radius* and can be interpreted as the radius at which the actual pressure equals the numerically computed pressure. The validity of the Peaceman well-index decreases rapidly with increasing near-well heterogeneity and grid skewness. It is also important to note that the Peaceman well-index is developed for the TPFA method and is not valid for other methods (such as MFE method with exact integration or mimetic methods in general) because different numerical methods in general give different well-cell pressures. We will now give a short description of how to extend Peaceman's results to other methods than TPFA; for a more extensive study of well models, we refer to [23].

Assuming steady-state, single-phase, horizontal flow in a homogeneous and isotropic reservoir with radial flow near the well, there exists an analytical flow

Table 2: The constant $C(\mathrm{IP}, \beta)$ in the formula (37) for r_0 for different mimetic inner products as a function of the aspect ratio β of the well cell.

β	1	2	4	8	16	32	64
IP_QFAMILY(t=1)	0.0495	0.0546	0.0655	0.0756	0.0823	0.0862	0.0883
IP_TPF	0.1404	0.1404	0.1404	0.1404	0.1404	0.1404	0.1404
IP_QFAMILY(t=3)	0.2015	0.1959	0.1857	0.1777	0.1729	0.1703	0.1690
IP_QFAMILY(t=4)	0.2423	0.2328	0.2153	0.2019	0.1939	0.1896	0.1874
IP_QFAMILY(t=5)	0.2712	0.2588	0.2363	0.2189	0.2087	0.2032	0.2003
IP_SIMPLE	0.2926	0.2782	0.2518	0.2316	0.2197	0.2133	0.2100

model near the well given by

$$p(r) = \frac{q\mu}{2\pi k \Delta z} \ln\Big(\frac{r}{r_w}\Big) + p_w. \tag{34}$$

We refer to [27, pages 150–153] for a derivation of this expression. Peaceman used the five-spot pattern problem to derive both numerical and analytical expressions for r_0 based on the relation above. However, the equivalent radius can also be calculated by simulating an infinite reservoir with one well and radial flow, and we employ this method to calculate new equivalent radii for mimetic methods.

The radial flow simulation is done by forcing appropriate flux boundary conditions. Consider a section of the reservoir containing a well as displayed in Figure 13 where we prescribe $F_i = q\theta_i/2\pi$ as flux boundary conditions on a face i. When the resulting system is solved for pressure and flow, we use the analytical pressure relation (34) with $r_w = r_0$ so that $p_w = p_E$, and set $r = r_b$ to be the distance from the well cell to the centroid of a boundary cell,

$$p(r_b) = \frac{q\mu}{2\pi k \Delta z} \ln\Big(\frac{r_b}{r_0}\Big) + p_E. \tag{35}$$

We then substitute $p(r_b)$ by p_b, the numerically computed pressure in the boundary cell (at r_b), and solve (35) for r_0,

$$r_0 = r_b \exp\Big(\frac{q\mu}{2\pi k \Delta z (p_E - p_b)}\Big). \tag{36}$$

If we assume that the equivalent radius for mimetic methods is on the same form as for TPFA shown in (33), then r_0 will depend on the actual inner product and on the grid aspect ratio of the well cell, $\beta = \Delta x/\Delta y$, i.e.,

$$r_0(\mathrm{IP}, \beta) = C(\mathrm{IP}, \beta)(\Delta x^2 + \Delta y^2)^{\frac{1}{2}}. \tag{37}$$

Thus, if we know r_0 we can easily compute $C(\mathrm{IP}, \beta)$, which is given in Table 2 as a function of β for the IP_TPF and IP_SIMPLE inner products and selected members of the IP_QFAMILY(t) inner-product family (27).

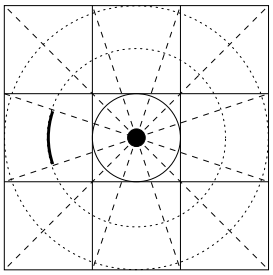


Figure 13: Section of infinite reservoir with well and radial flow.

4 Multiscale Pressure Solvers

A main purpose of MRST is to provide an efficient toolkit for the development and testing of new ideas. In this section, we present examples in which the solvers described in the previous sections are applied as a building block to develop multiscale pressure solvers. Given the importance of grid geometry on the quality of numerical computations, it is crucial to have flexible tools that allow testing new algorithms on many different types of polyhedral grids of varying complexity in 3D. Through the discussion of multiscale solvers, we hope to give the reader a taste of the utility of the toolbox.

Multiscale flow solvers [17, 15] 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 of multiscale methods is the incorporation of fine-scale information into a set of coarse-scale equations in a way that is consistent with the local property of the differential operator. Generally, 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. In particular, the compression operator reduces 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 constructs 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 [18, 32], the coarse-scale degrees-of-freedom are associated with pressure values at the vertices 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 [11, 4],

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

$$\vec{\psi}_{ij} = -\mathbf{K}\nabla p_{ij}, \qquad \nabla\cdot\vec{\psi}_{ij} = \begin{cases} w_i(x), & \text{if } x\in\Omega_i,\\ -w_j(x), & \text{if } x\in\Omega_j,\\ 0, & \text{otherwise,}\end{cases} \tag{38}$$

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 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 [2] 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

$$\begin{bmatrix} \boldsymbol{\Psi}^{\mathsf{T}}\boldsymbol{B}\boldsymbol{\Psi} & \boldsymbol{\Psi}^{\mathsf{T}}\boldsymbol{C}\boldsymbol{\mathcal{I}} & \boldsymbol{\Psi}^{\mathsf{T}}\boldsymbol{D}\boldsymbol{\mathcal{J}}\\ \boldsymbol{\mathcal{I}}^{\mathsf{T}}\boldsymbol{C}^{\mathsf{T}}\boldsymbol{\Psi} & \mathbf{0} & \mathbf{0}\\ \boldsymbol{\mathcal{J}}^{\mathsf{T}}\boldsymbol{D}^{\mathsf{T}}\boldsymbol{\Psi} & \mathbf{0} & \mathbf{0}\end{bmatrix}\begin{bmatrix}\boldsymbol{u}^c\\ -\boldsymbol{p}^c\\ \boldsymbol{\pi}^c\end{bmatrix} = \begin{bmatrix}\mathbf{0}\\ \boldsymbol{\mathcal{I}}^{\mathsf{T}}\boldsymbol{q}\\ \mathbf{0}\end{bmatrix}. \tag{39}$$

Once (39) 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 14. More details about how to include wells in a appropriate way is given in [31].

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

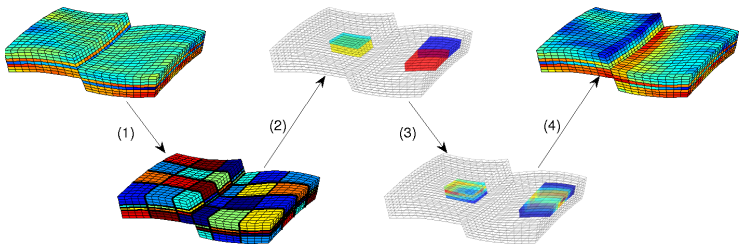


Figure 14: Key steps of the multiscale method: (1) blocks in the coarse grid are defined as connected collections of cells from the fine grid; (2) a local flow problem is defined for all pairs of blocks sharing a common face; (3) the local flow problems are solved and the solutions are collected as basis functions (reconstruction operators); and (4) the global coarse-system (39) is assembled and solved, then a fine-scale solution can be reconstructed.

Example 5 (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, 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 [4]. 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);
```

The permeability and the coarse grid are shown in Figure 15. 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 15. 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

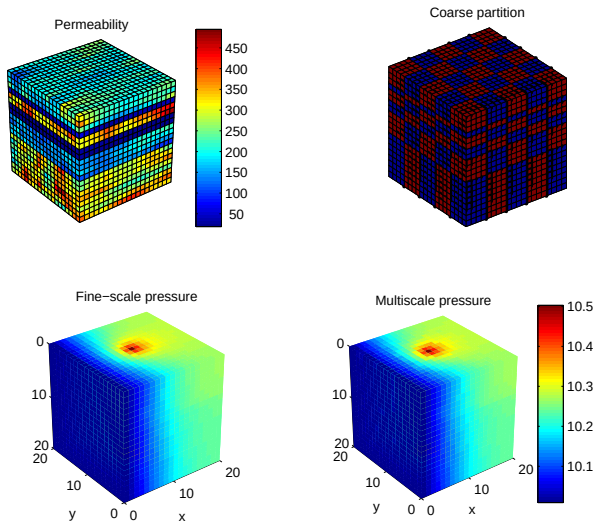


Figure 15: Comparison of the MsMFE and fine-scale mimetic flow solvers for the setup from Figure 4. The top plots show the layered permeability field and the corresponding partitioning into coarse blocks. The lower plots show the fine-scale pressure solution (left) and the multiscale solution (right).

solvers for small problems and e.g., the AGMG solver [29] for large problems. The advantage of a multiscale solver comes first when considering multiphase flow problems, where the key to computational efficiency is reuse of previous computations. For a multiphase system, the basis functions will become time-dependent when $\mathbf{K}$ is replaced by $\lambda\mathbf{K}$ in (38), 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 (39) is usually inexpensive compared to solving the corresponding fine-scale system or computing basis functions. In the current release of MRST, gravity is only resolved on the coarse scale. However, gravity effects can easily be included also on the fine scale by using special correction functions.

Next, we present a short example that 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 6 (Prismatic and 2.5D PEBI Grids) *The MsMFE method was previously demonstrated on Cartesian and logically Cartesian (corner-point) grids, see [3, 20, 4]. However, the method is equally feasible on general, unstructured grids provided there exists a suitable partitioning of the underlying (fine-scale) grid. Figure 16 shows the solution of the single-phase flow problem* (1) *with isotropic, homogeneous* $\mathbf{K}$ *on a computational domain* Ω *that has been*

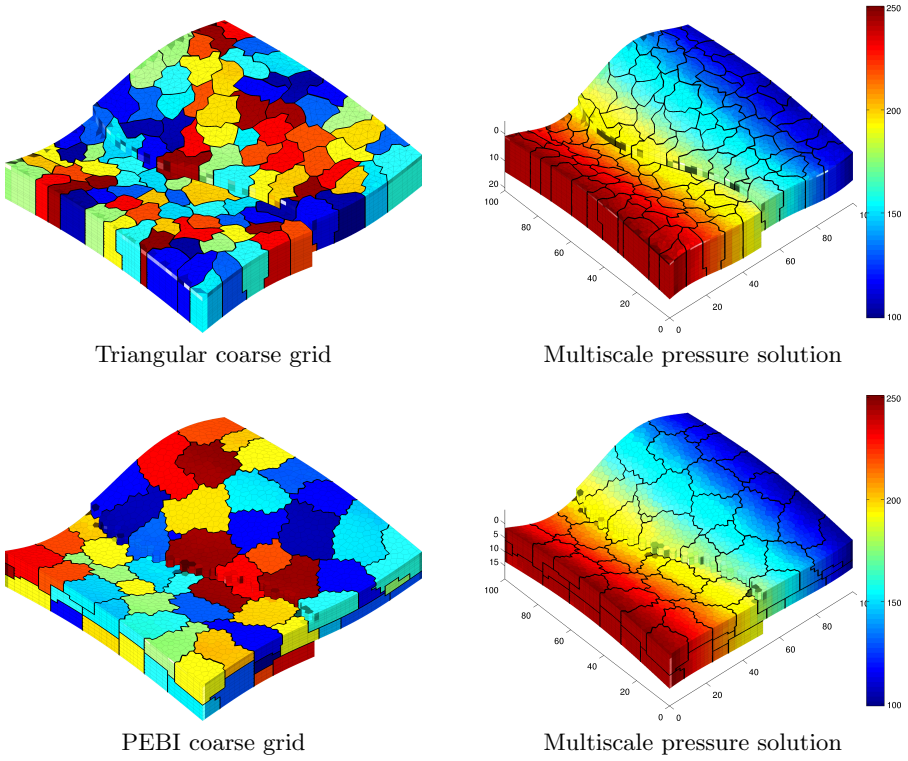


Figure 16: Multiscale discretisation of a flow problem with linear pressure drop on a prismatic and a PEBI grid.

discretised using both prismatic and PEBI cells. These cells are subsequently partitioned into a coarse grid by means of the well-known software package METIS [19]. For simplicity, the areal PEBI grid was created as the Voronoi diagram of the triangular grid used to extrude the prismatic grid and hence the fault is more irregular on the PEBI 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 on both grids.

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 last 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 7 (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*

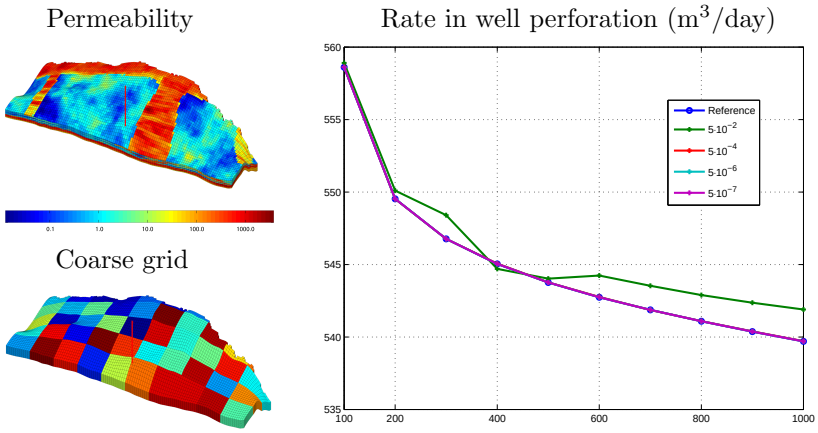


Figure 17: Primary production from a gas reservoir computed by the MsMFE method.

the shallow-marine realisations from the SAIGUP study [25]. One approach for simulating compressible black-oil models with the MsMFE method, is to use a mixed residual formulation

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

Here, elliptic multiscale basis functions computed from (38) *using a weight function $w_i(x)$ that scales like the porosity act as predictors. Compressibility effects are accounted for by a parabolic correction that 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 17) and is assumed to be initially filled with an ideal gas at* 200 bar *pressure. The reservoir is produced by a single producer, operating at a constant bottom-hole pressure of* 150 bar*. Figure 17 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.*

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 discretisation

for the pressure equation and an implicit version of the scheme (7) for saturation. In addition, an adjoint code for the multiscale method combined with the flow based coarsening approach is included, see [22].

Example 8 (Optimising Net Present Value (NPV)) *In this example we consider a synthetic model with two horizontal injectors and one multilateral producer, see Figure 18. 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 PVI*. In Figure 18 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.*

5 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 (MRST) 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. In this paper we have given an overview of MRST and discussed in detail a family of consistent methods that are convergent on fully unstructured, polyhedral grids. We have also demonstrated the compressible black-oil and adjoint multiscale features that currently exists only

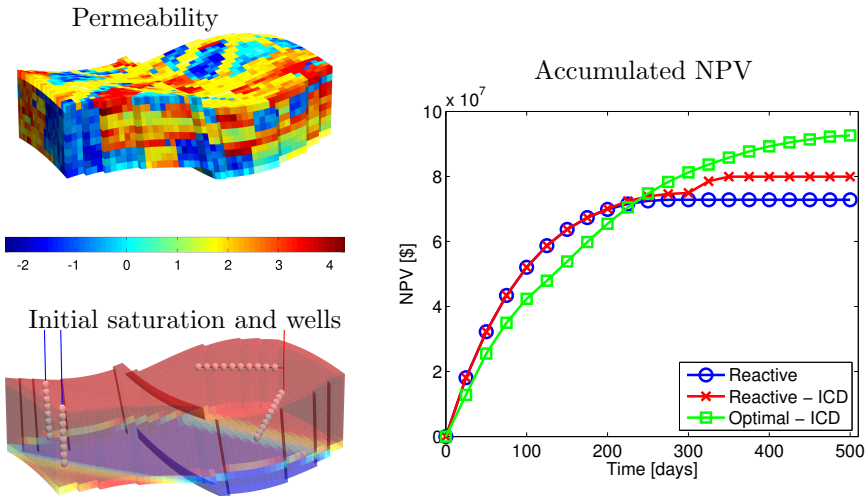


Figure 18: Accumulated net present value for the three production strategies.

in our in-house, development version of the package. 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.

6 Acknowledgements

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

References

[1] Aarnes, J.E., Gimse, T., Lie, K.A.: An introduction to the numerics of flow in porous media using Matlab. In: G. Hasle, K.A. Lie, E. Quak (eds.) Geometrical Modeling, Numerical Simulation and Optimisation: Industrial Mathematics at SINTEF, pp. 265–306. Springer Verlag, Berling Heidelberg New York (2007)

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

[3] Aarnes, J.E., Krogstad, S., Lie, K.A.: 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) (2006)

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

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

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

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

[8] Brezzi, F., Fortin, M.: Mixed and Hybrid Finite Element Methods, *Springer Series in Computational Mathematics*, vol. 15. Springer-Verlag, New York (1991)

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

[10] Chavent, G., Jaffre, J.: Mathematical models and finite elements for reservoir simulation. North Holland (1982)

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

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

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

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

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

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

[17] Hou, T., Wu, X.H.: A multiscale finite element method for elliptic problems in composite materials and porous media. J. Comput. Phys. **134**, 169–189 (1997)

[18] Jenny, P., Lee, S.H., Tchelepi, H.A.: Multi-scale finite-volume method for elliptic problems in subsurface flow simulation. J. Comput. Phys. **187**, 47–67 (2003)

[19] Karypis, G., Kumar, V.: A fast and high quality multilevel scheme for partitioning irregular graphs. In: International Conference on Parallel Processing, pp. 113–122 (1995)

[20] Kippe, V., Aarnes, J.E., Lie, K.A.: A comparison of multiscale methods for elliptic problems in porous media flow. Comput. Geosci. **12**(3), 377–398 (2008). DOI 10.1007/s10596-007-9074-6

[21] Klausen, R.A., Stephansen, A.F.: Mimetic MPFA. In: Proc. 11th European Conference on the Mathematics of Oil Recovery, 8-11 Sept., Bergen, Norway, A12. EAGE (2008)

[22] Krogstad, S., Hauge, V.L., Gulbransen, A.F.: Adjoint multiscale mixed finite elements. SPE J. **16**(1), 162–171 (2011). DOI 10.2118/119112-PA

[23] Ligaarden, I.S.: Well Models for Mimetic Finite Difference Methods and Improved Representation of Wells in Multiscale Methods. Master Thesis, University of Oslo (2008)

[24] Lipnikov, K., Shashkov, M., Yotov, I.: Local flux mimetic finite difference methods. Numer. Math. **112**(1), 115–152 (2009). DOI 10.1007/s00211-008-0203-5

[25] Manzocchi, T.., et al.: 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 (2008)

[26] The MATLAB Reservoir Simulation Toolbox, version 2011a (2011). URL `http://www.sintef.no/MRST/`

[27] Muskat, M.: The flow of homogeneous fluid through porous media. McGraw Hill Book Co., Inc, New York (1937)

[28] Natvig, J.R., Lie, K.A.: Fast computation of multiphase flow in porous media by implicit discontinuous Galerkin schemes with optimal ordering of elements. J. Comput. Phys. **227**(24), 10,108–10,124 (2008). DOI 10.1016/j.jcp.2008.08.024

[29] Notay, Y.: An aggregation-based algebraic multigrid method. Electronic Transactions on Numerical Analysis **37**, 123–146 (2010)

[30] Peaceman, D.W.: Interpretation of well-block pressures in numerical reservoir simulation with nonsquare grid blocks and anisotropic permeability. SPE, Trans. AIME **275**, 10–22 (1983)

[31] Skaflestad, B., Krogstad, S.: Multiscale/mimetic pressure solvers with near-well grid adaption. In: Proceedings of ECMOR XI–11th European Conference on the Mathematics of Oil Recovery, A36. EAGE, Bergen, Norway (2008)

[32] Zhou, H., Tchelepi, H.: Operator-based multiscale method for compressible flow. SPE J. **13**(2), 267–273 (2008)