softwareunderground.org presents

TRANSFORM 2021

Virtual Conference on the Digital Subsurface, 16–23 April

supported by

aws STUDIO X Agile*

Welcome! The livestream will begin soon...

Reservoir Simulation and Modelling with MRST

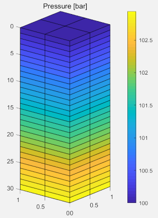
Øystein S. Klemetsdal

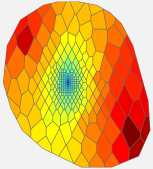
Computational Geosciences, SINTEF Digital, Oslo, Norway

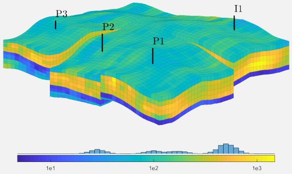
TRANSFORM 2021, Software underground, April 20, 2021
Slack channel: `t21-tue-mrst`

Overview of the tutorial

- Short overview of MRST
  - what is the purpose of this software?
  - how is the software organized?
  - where can I find help?
- Getting started
  - download or clone MRST
  - using and navigating the modules
  - solving an incompressible flow problem
- Numerical framework
  - discrete operators and automatic differentiation
- More complex example
  - creating and simulating a sector model







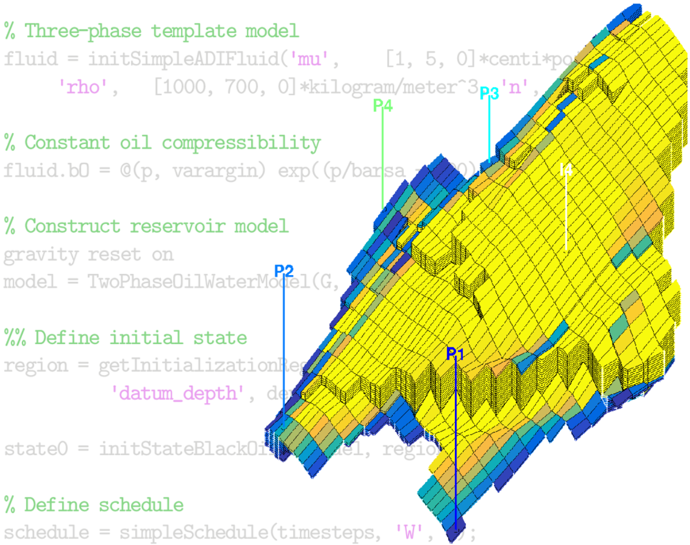
Short overview of MRST

MATLAB Reservoir Simulation Toolbox (MRST)

Transforming research on reservoir modelling

Unique prototyping platform:

- Standard data formats
- Data structures/library routines
- Fully unstructured grids
- Rapid prototyping:
  - discrete operators
  - automatic differentiation
  - object-oriented framework
- Industry-standard simulation



http://www.mrst.no

MATLAB Reservoir Simulation Toolbox (MRST)

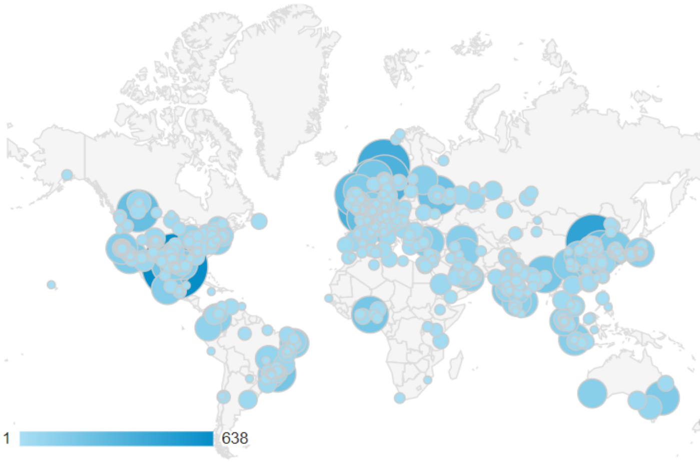
Transforming research on reservoir modelling

Large international user base:

- downloads from the whole world
- 125 master theses
- 62 PhD theses
- 270 journal papers (not by us)
- 150 proceedings papers

Numbers are from Google Scholar notifications

Used both by academia and industry



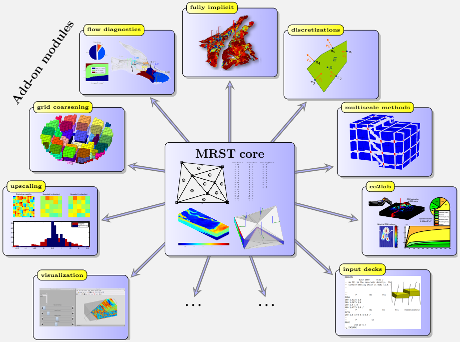
Google Analytics: access pattern for `www.mrst.no`
Period: 1 July 2018 to 31 December 2019

Community code: software organization

Modular design:

- **small core** with mature and well-tested functionality used in *many* programs or modules
- **semi-independent modules** extend core functionality
- in-source documentation like in MATLAB
- all modules must have code examples and/or tutorials

This simplifies how we distinguish public and in-house or client-specific functionality



Community code: software organization

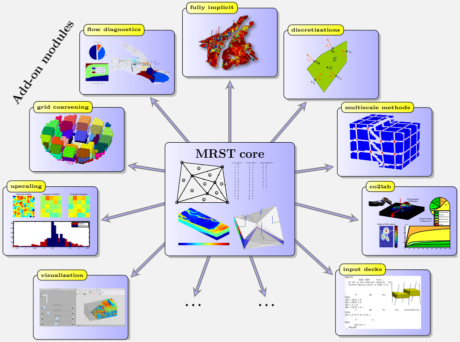
Modular design:

- **small core** with mature and well-tested functionality used in *many* programs or modules
- **semi-independent modules** extend core functionality
- in-source documentation like in MATLAB
- all modules must have code examples and/or tutorials

This simplifies how we distinguish public and in-house or client-specific functionality

Core module:

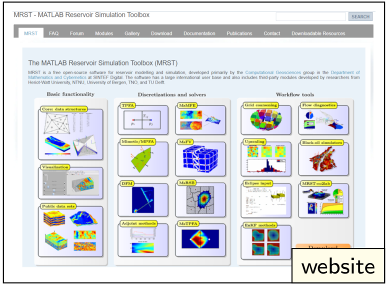
grid structure, grid factory routines, petrophysical data, basic fluid models, automatic differentiation library, setting boundary/wells/sources, reservoir state, visualization, etc



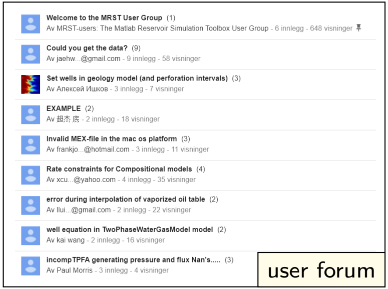
Latest release: 57 modules

- Grid generation and coarsening
- ECLIPSE input and output
- Upscaling / multiscale solvers
- Consistent discretizations
- Black-oil, EOR, compositional
- Fractures: DFM, EDFM, DPDP
- Geomechanics, geochemistry, geothermal
- Unsaturated media (Richards eq.)
- Multisegment wells (general network)
- CO2 storage laboratory
- Adjoints, optimization, ensembles
- Pre/postprocessing/visualization
- Flow diagnostics
- ...

User resources (getting help)

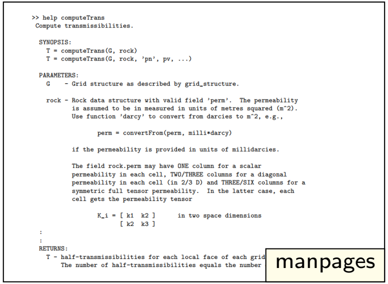


website

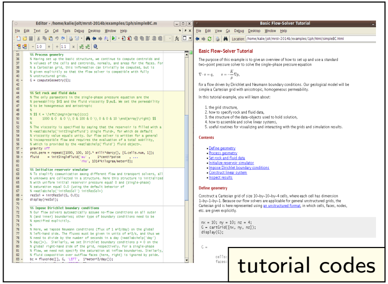


user forum

textbook



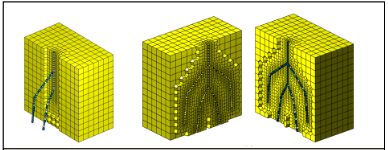
manpages



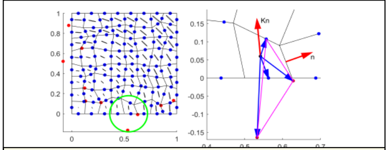
tutorial codes

online tutorials

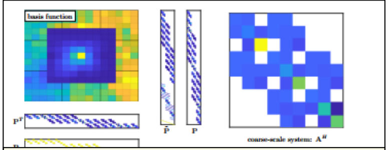
New book: Advanced modeling with MRST



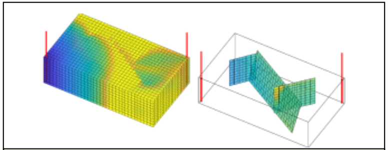
Berge et al.: Constrained Voronoi grids



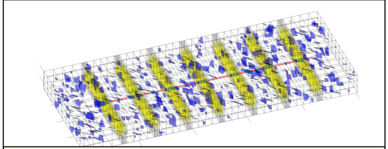
Al Kobaisi & Zhang: nonlinear FVM



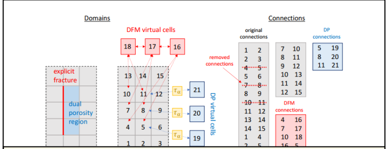
Lie & Møyner: multiscale methods



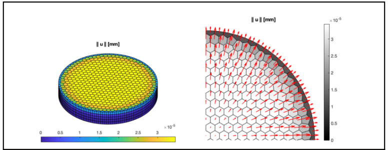
Wong et al.: embedded discrete fractures



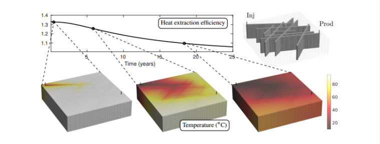
Olorode et al.; fractured unconventionals



March et al.: unified framework, fractures



Varela et al.: unsaturated poroelasticity



Collignon et al.: geothermal systems

Klemetsdal & Lie: discontinuous Galerkin

Møyner: state functions, AD backends

Sun et al.: chemical EOR

Møyner: compositional

Andersen: coupled flow & geomechanics

Software requirements

Minimal requirement is MATLAB version 7.4 (R2007a). No toolboxes!

Certain modules use features that were not present in R2007a:

- Automatic differentiation relies upon new-style classes (classdef) from R2008a.
- Various scripts and examples use new syntax for random numbers from R2007b.
- Some scripts use tilde operator from R2009b to ignore return values.
- Some solvers (e.g., fully implicit) are not efficient on versions older than R2011b.

Most of MRST can be used with GNU Octave, maybe except for some GUIs. The AD-OO solvers are somewhat slow, but will get better performance in the next release.

External dependencies:

- `AGMG` or `AMGCL` for iterative linear solvers (multigrid, etc)
- `MATLAB-BGL` (MATLAB Boost Graph Library) for graph algorithms
- `METIS` for partitioning of fully unstructured grids, etc.

Getting started

Getting the software

- URL: `www.mrst.no`
- Free software with GNU GPL license
- Released twice per year
- Latest release: MRST 2021a, from 19/04/2021 (yesterday!)
- Provided as a self-contained archive file (e.g., `mrst-2021a.zip`)
- We recommend that you join the `MRST-announce` Google group

Follow development actively:
`bitbucket.org/mrst/mrst-core`

Installing MRST

Unzip the software to a subfolder mrst-2021a of your current working directory:

```
unzip('mrst-2021a.zip')
```

Once MRST is extracted to a directory, you must navigate MATLAB there. On Linux/Mac OS,

```
cd /home/username/mrst-2021a/
```

or on Windows,

```
cd C:\Users\username\mrst-2021a\
```

assuming that the files were extracted to the home directory. The startup.m file must then be run to activate MRST,

```
startup;
```

or you can call the startup script directly

```
run /home/username/mrst-2021a/startup
```

If you start MATLAB in the directory containing MRST, or run the `startup.m` file, you will see the following message

Command Window

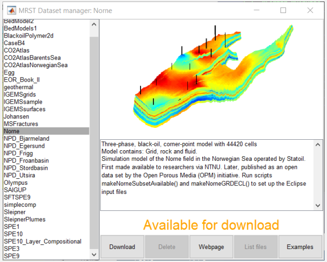
```
Welcome to the Matlab Reservoir Simulation Toolbox (MRST)!
You are using the release version 2020b. To download other versions of MRST
and view examples and relevant publications, please visit www.mrst.no

Useful commands for getting started:
 - List all introductory examples:    mrstExamples()
 - List all modules:                  mrstPath('list')
 - Load modules using GUI:            mrstModule('gui')
 - Explore all available data sets    mrstDatasetGUI()
 - List examples of a module:         mrstExamples('ad-blackoil')
 - Explore modules and publications:  mrstExploreModules()
 - Show all examples in all modules:  mrstExamples('all')
 - Display this message:              mrstStartupMessage()

For assistance and discussions about MRST, please visit our mailing list at
    www.sintef.no/projectweb/mrst/forum/ (sintef-mrst@googlegroups.com)
For some common queries, see our FAQ: www.sintef.no/projectweb/mrst/faq/
fx >>
```

The core module of MRST offers a number of examples that introduce you to data structures and data sets, how to set up basic solvers, how to visualize input data and simulation results, etc

```
>> mrstExamples
Module "core" has 14 examples:
    flowSolverTutorial1.m
    flowSolverTutorialAD.m
    tutorialAD.m
    tutorialBasicObjects.m
    tutorialPlotting.m
    datasets\showCaseB4.m
    datasets\showJohansen.m
    datasets\showNorne.m
    datasets\showSAIGUP.m
    datasets\showSPE10.m
    grids\gridTutorialCornerPoint.m
    grids\gridTutorialIntro.m
    grids\gridTutorialStruct.m
    grids\gridTutorialUnstruct.m
```

We go through the following:

- Visit the website, forum, and FAQ
- List basic tutorials, `mrstExample()`
- Bring up `mrstExploreModules()`
- Bring up `mrstDatasetGUI()`



Operating modules

Graphical user interface to modules:

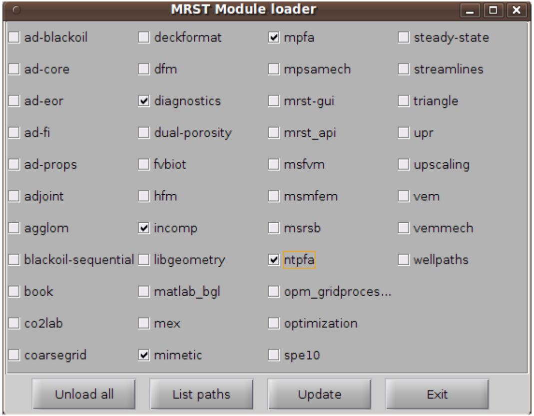
```
mrstModule('gui')
moduleGUI
```

List all modules and their path

```
mrstPath
```

Load new modules

```
mrstModule add mimetic mpfa
```



Adding your own modules

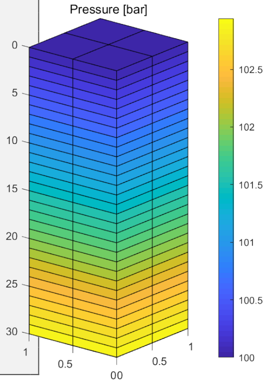
```
mrstPath reregister distmesh ...
    /home/username/mrst-2016b/utils/3rdparty/distmesh
```

Incompressible flow solvers

```
% Activate module for incompresisble solvers
mrstModule add incomp

%% Define the model
gravity reset on
G     = cartGrid([2, 2, 30], [1, 1, 30]);
G     = computeGeometry(G);
rock  = makeRock(G, 0.1*darcy, 1);
fluid = initSingleFluid('mu', 1*centi*poise, ...
              'rho', 1014*kilogram/meter^3);
bc    = pside([], G, 'TOP', 100.*barsa());

%% Assemble and solve the linear system
T   = computeTrans(G, rock);
sol = incompTPFA(initResSol(G, 0.0), ...
                 G, T, fluid, 'bc', bc);

%% Plot the face pressures
newplot;
plotFaces(G, 1:G.faces.num, sol.facePressure./barsa);
set(gca,'ZDir','reverse','DataAspectRatio',[1 1 10])
title('Pressure [bar]')
view(3), colorbar
```

Code: `flowSolverTutorial1.m`

$$\nabla \cdot \nabla(p + \rho\vec{g}) = 0$$



Will go through this code in detail.

Earliest parts of MRST:

- Procedural programming
- Structs for reservoir state, rock parameters, wells, b.c., and source term
- Fluid behavior: struct with function pointers

Advantages:

- **hide specific details** of geomodel and fluid model
- **vectorization**: efficient/compact code
- **unified access** to key parameters

softwareunderground.org presents

TRANSFORM 2021

Virtual Conference on the Digital Subsurface, 16–23 April

supported by

Taking a break... back soon.

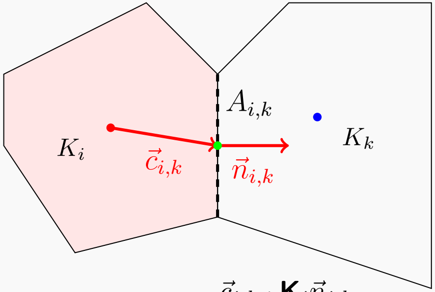
Numerical framework

Rapid prototyping: discrete differentiation operators

Conservation of momentum (Darcy's law):

$$\int_{\Gamma_f} \vec{v}(x) \cdot \vec{n}_f \, ds = - \int_{\Gamma_f} \mathbf{K}(x) \nabla p \cdot \vec{n}_f \, ds$$

Two-point flux approximation



$$T_{i,k} = A_{i,k} \frac{\vec{c}_{i,k} \cdot \mathbf{K}_i \vec{n}_{i,k}}{|\vec{c}_{i,k}|^2}$$

$$T_{ik} = [T_{i,k}^{-1} + T_{k,i}^{-1}]^{-1}$$

$$v_{ik} = T_{ik}(p_i - p_k)$$

Rapid prototyping: discrete differentiation operators

Conservation of momentum (Darcy's law):

$$\int_{\Gamma_f} \vec{v}(x) \cdot \vec{n}_f \, ds = -\int_{\Gamma_f} \mathbf{K}(x) \nabla p \cdot \vec{n}_f \, ds$$

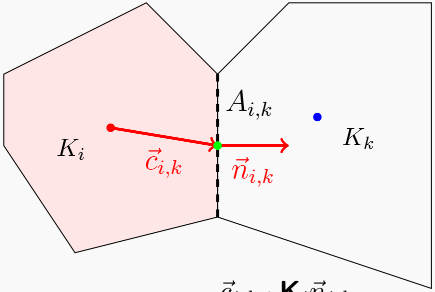
discrete: $\boldsymbol{v}[f] = -\boldsymbol{T}[f]\,\mathtt{grad}(\boldsymbol{p})[f]$

Conservation of mass:

$$\int_{\partial\Omega_c} \vec{v} \cdot \vec{n} \, ds = \int_{\Omega_c} \nabla \cdot \vec{v} \, d\vec{x} = \int_{\Omega_c} q d\vec{x}$$

discrete: $\mathtt{div}(\boldsymbol{v})[c] = \boldsymbol{q}[c]$

Two-point flux approximation



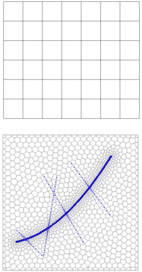
$$T_{i,k} = A_{i,k} \frac{\vec{c}_{i,k} \cdot \mathbf{K}_i \vec{n}_{i,k}}{|\vec{c}_{i,k}|^2}$$

$$T_{ik} = [T_{i,k}^{-1} + T_{k,i}^{-1}]^{-1}$$

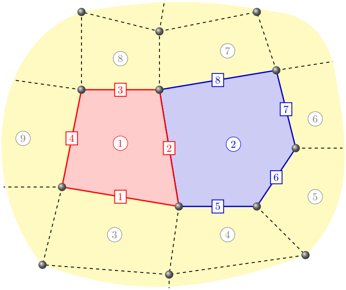
$$v_{ik} = T_{ik}(p_i - p_k)$$

Rapid prototyping: discrete differentiation operators

Idealized models



Grid structure in MRST



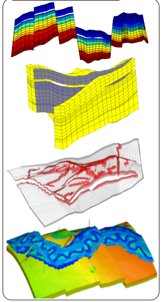
c	F(c)
1	1
1	2
1	3
1	4
2	5
2	6
2	7
2	8
2	2
3	1
⋮	⋮
⋮	⋮

Map: cell → faces

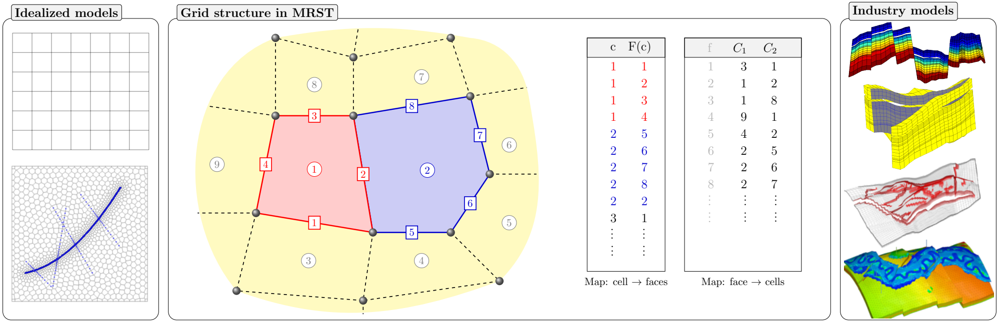
f	C_1	C_2
1	3	1
2	1	2
3	1	8
4	9	1
5	4	2
6	2	5
7	2	6
8	2	7
⋮	⋮	⋮
⋮	⋮	⋮

Map: face → cells

Industry models



Rapid prototyping: discrete differentiation operators



For finite volumes, the discrete `grad` operator maps from cell pair $C_1(f), C_2(f)$ to face f:

$$\texttt{grad}(\boldsymbol{p})[f] = \boldsymbol{p}[C_2(f)] - \boldsymbol{p}[C_1(f)],$$

where $\boldsymbol{p}[c]$ is a scalar quantity associated with cell c. Discrete `div` maps from faces to cells.

Both are linear operators and can be represented as sparse matrix multiplications.

Close correspondence with mathematics

Continuous

Incompressible flow:

$$\nabla \cdot (\mathbf{K}\nabla p) + q = 0$$

Compressible flow:

$$\frac{\partial(\phi\rho)}{\partial t} + \nabla \cdot (\rho\mathbf{K}\nabla p) + q = 0$$

Discrete in MATLAB

Incompressible flow:

```
eq = div(T .* grad(p)) + q;
```

Compressible flow:

```
eq = (pv(p).*rho(p)-pv(p0).*rho(p0))/dt ...
    + div(avg(rho(p)).*T.*grad(p))+q;
```

Close correspondence with mathematics

Continuous

Incompressible flow:

$$\nabla \cdot (\mathbf{K}\nabla p) + q = 0$$

Compressible flow:

$$\frac{\partial(\phi\rho)}{\partial t} + \nabla \cdot (\rho\mathbf{K}\nabla p) + q = 0$$

Discrete in MATLAB

Incompressible flow:

```
eq = div(T .* grad(p)) + q;
```

Compressible flow:

```
eq = (pv(p).*rho(p)-pv(p0).*rho(p0))/dt ...
    + div(avg(rho(p)).*T.*grad(p))+q;
```

Discretization of flow models leads to large systems of nonlinear equations. Can be linearized and solved with Newton's method

$$\boldsymbol{F}(\boldsymbol{u}) = \mathbf{0} \qquad \Rightarrow \qquad \frac{\partial \boldsymbol{F}}{\partial \boldsymbol{u}}(\boldsymbol{u}^i)(\boldsymbol{u}^{i+1} - \boldsymbol{u}^i) = -\boldsymbol{F}(\boldsymbol{u}^i)$$

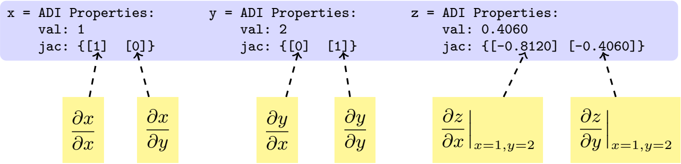
Coding necessary Jacobians is time-consuming and error prone

Rapid prototyping: automatic differentiation

General idea:

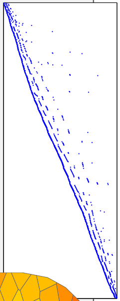
- Any code consists of a limited set of arithmetic operations and elementary functions
- Introduce an extended pair, $\langle x, 1\rangle$, i.e., the value x and its derivative 1
- Use chain rule and elementary derivative rules to mechanically accumulate derivatives *at specific values* of x
  - Elementary: $v = \sin(x) \longrightarrow \langle v\rangle = \langle \sin x, \cos x\rangle$
  - Arithmetic: $v = fg \longrightarrow \langle v\rangle = \langle fg, fg_x + f_x g\rangle$
  - Chain rule: $v = \exp(f(x)) \longrightarrow \langle v\rangle = \langle \exp(f(x)), \exp(f(x))f'(x)\rangle$
- Use operator overloading to avoid messing up code

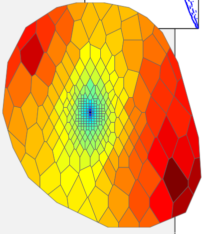
```
[x,y] = initVariablesADI(1,2);
z = 3*exp(-x*y)
```



Solving the Poisson equation: unstructured grid


```
% Grid and grid information
load seamount
G    = pebi(triangleGrid([x(:) y(:)]));
G    = computeGeometry(G);
rock = makeRock(G, 1, 1);
nc   = G.cells.num;

% Operators
S = setupOperatorsTPFA(G,rock); spy(S.C);

% Assemble and solve equations
p = initVariablesADI(zeros(nc,1));
q = zeros(nc, 1)
q([135 282 17]) = [-1 .5 .5];

eq    = S.Div(S.T.*S.Grad(p))+q;
eq(1) = eq(1) + p(1);
p     =-eq.jac{1}\eq.val;
plotCellData(G,p);
```





Discretization of $-\nabla \cdot (\mathbf{K}\nabla p) = q$ gives the residual flow equation

$$\mathbf{0} = \mathtt{div}(\boldsymbol{T}\mathtt{grad}(\mathbf{p})) + \mathbf{q} = \mathbf{F}(\mathbf{p})$$

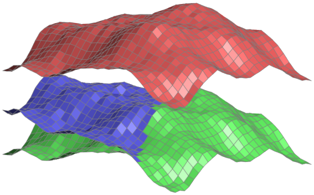
Automatic differentiation gives us $\partial\mathbf{F}/\partial\mathbf{p}$

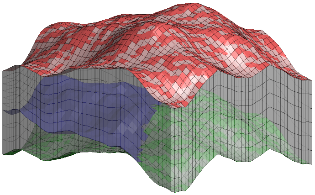
Go through the code in detail:

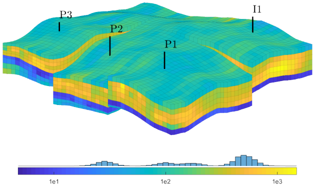
- Explain the gridding
- Show the grid structure
- Look at the operators
- Look at Jacobians, etc

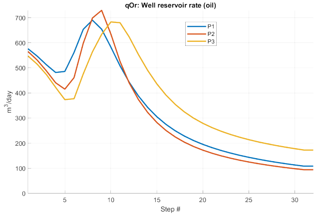
More complex example

Example: from geological horizons to simulation









Worked simulation example:

- Define horizons
- Extrude grid to mimic stratigraphy
- Introduce structural architecture
- Add petrophysics
- Define wells
- Setup fluid model
- Initial state
- Simulation schedule (well controls)
- Simulate with computational steering

softwareunderground.org presents

TRANSFORM 2021

Virtual Conference on the Digital Subsurface, 16–23 April

supported by

aws STUDIO X Agile*

The livestream is over. Thanks for taking part!