Recent Development in MRST

Bård Skaflestad

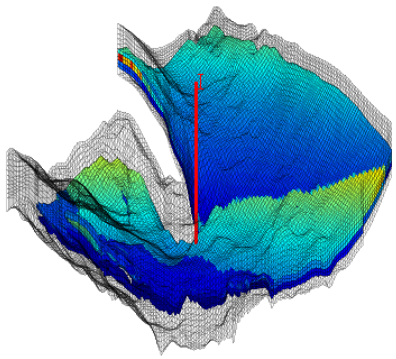
SINTEF ICT, Applied Mathematics

OPM Symposium, Bergen, 28–29 May 2013

The MATLAB Reservoir Simulation Toolbox (MRST)

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

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



Download

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

Version 2013a was released on the 19th of April, 2013, and can be downloaded under the terms of the GNU General Public License (GPL)

How is MRST Designed?

The fundamental object in MRST is the grid:

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

Physical quantities defined as dynamic objects (structures) in MATLAB

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

All MRST operations accept, manipulate and produce objects of these types.
Physical quantities are assumed to be in SI units ($[\mathbf{K}] = \mathrm{m}^2$, $[\mu] = \mathrm{Pa} \cdot \mathrm{s}$ etc).

How is MRST Designed?

MRST Core

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

Functionality is stable and not expected to change in future releases

Modules

Add-on software that extends, complements, and overrides existing MRST features. Presently implements more advanced solvers and tools:

- automatic differentiation, inexpensive flow diagnostics
- adjoint methods, black-oil models, vertically integrated models, ...

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

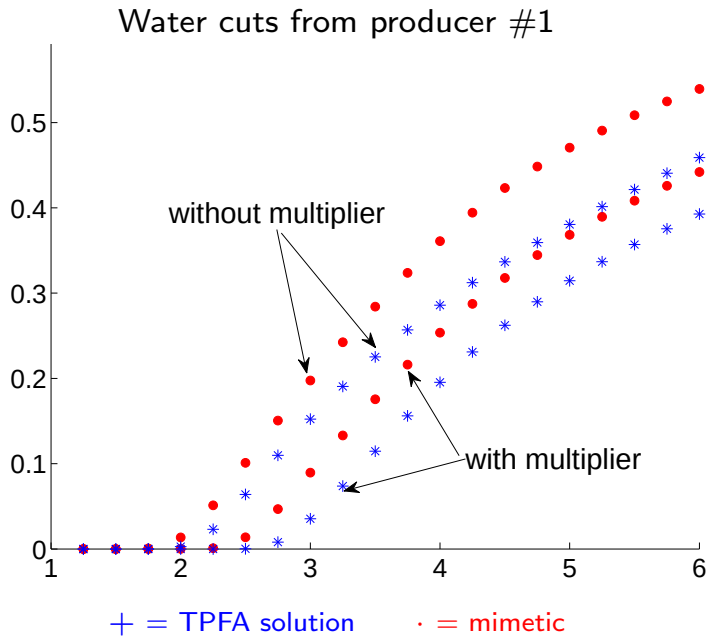
Application Examples

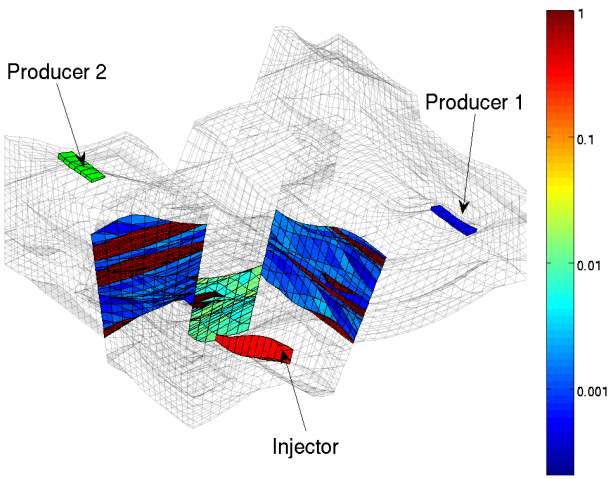
Modelling Faults in Consistent Schemes

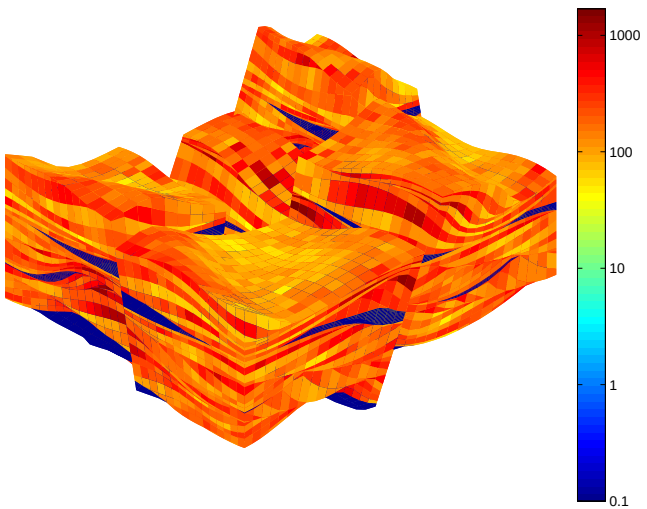
Faults modelled as internal boundaries, with internal jump conditions

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

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





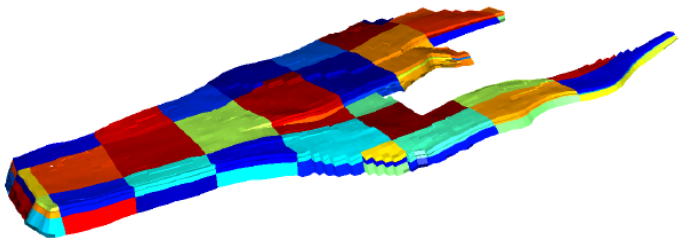


Application Examples

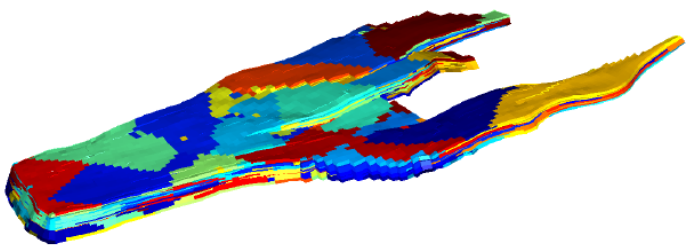
Production optimisation

Specialised simulator: using different grids for pressure and transport

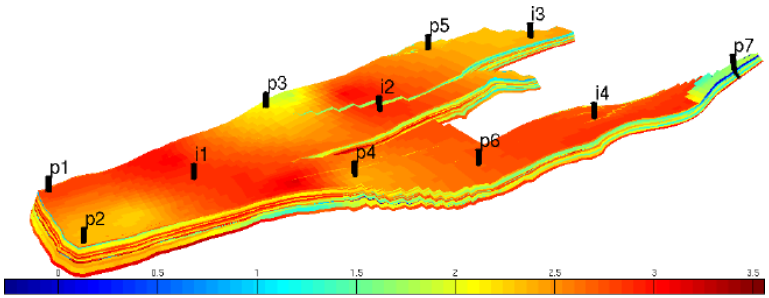
Multiscale Pressure Solver



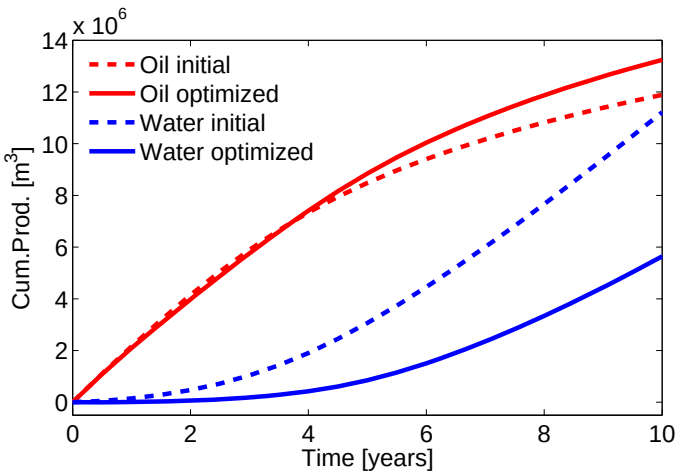
Transport on Flow-adapted Grid



Water-flood Optimisation



Reservoir geometry from a Norwegian Sea field



Forward simulations:
44 927 cells, 20 time steps, $<$ 5 sec in MATLAB

Automatic Differentiation

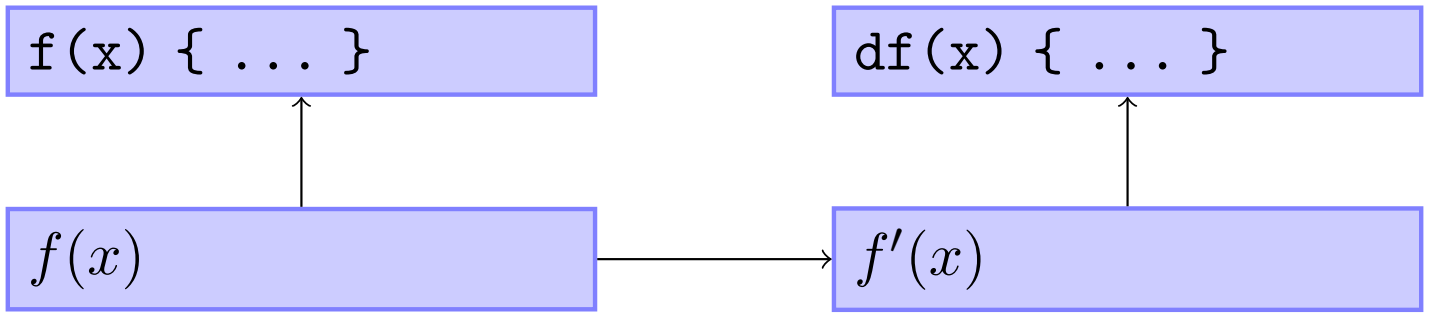
```
f(x) { ... }
```

```
df(x) { ... }
```

$f(x)$

$f'(x)$

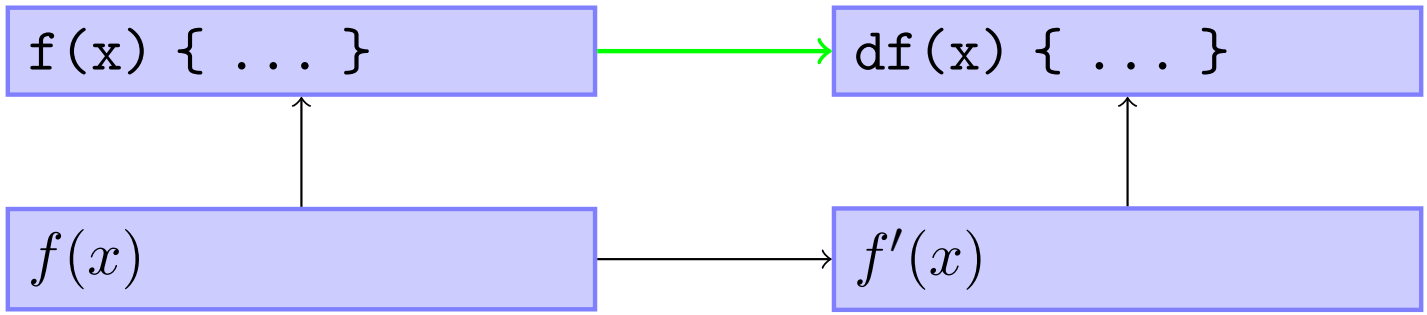
Automatic Differentiation



Traditional Process

- Human implements code to evaluate $f(x)$
- Manual or symbolic calculation to derive $f'(x)$
- Human implements code to evaluate $f'(x)$

Automatic Differentiation



Traditional Process

- Human implements code to evaluate $f(x)$
- Manual or symbolic calculation to derive $f'(x)$
- Human implements code to evaluate $f'(x)$

Automatic Differentiation

- Human implements code to evaluate $f(x)$
- Computer code to evaluate $f'(x)$ is automatically generated

Fully Implicit Solvers in MRST Based on AD

Black-oil Mass Balance Equations (w/dissolved gas)

$$\begin{cases} \partial_t(\phi b_w s_w) + \operatorname{div}(b_w v_w) = q_w \\ \partial_t(\phi b_o s_o) + \operatorname{div}(b_o v_o) = q_o \\ \partial_t(\phi(b_g s_g + R_s b_o s_o)) + \operatorname{div}(b_g v_g + R_s b_o v_o) = q_g \end{cases}$$

Fully Implicit Solvers in MRST Based on AD

Black-oil Mass Balance Equations (w/dissolved gas)

$$\begin{cases} \partial_t(\phi b_w s_w) + \operatorname{div}(b_w v_w) = q_w \\ \partial_t(\phi b_o s_o) + \operatorname{div}(b_o v_o) = q_o \\ \partial_t(\phi(b_g s_g + R_s b_o s_o)) + \operatorname{div}(b_g v_g + R_s b_o v_o) = q_g \end{cases}$$

MRST Implementation; p_o, s_w, s_g and R_s as Primary Degrees of Freedom

```
F{w} = (pv/dt).*(bW.*sW         - bW(p0).*sW0) + div(bWvW);
F{o} = (pv/dt).*(bO.*(1-sW-sG) - bO(p0,rs0).*(1-sW0-sG0)) + div(bOvO);
F{g} = (pv/dt).* ...
       ( bG.*sG      + rs.*bO.*(1-sW-sG) - ...
         bG(p0).*sG0 + rs0.*bO(p0,rs0).*(1-sW0-sG0) ) + ...
       div(bGvG + rsbOvO)
```

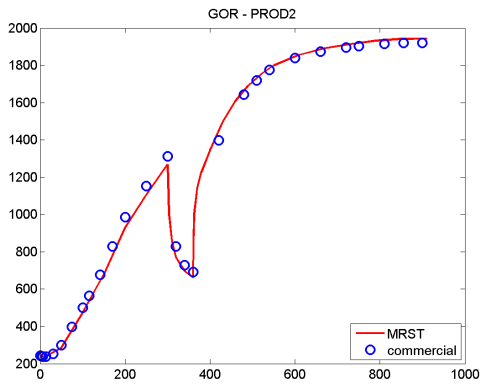
Residuals in `F{i}.val` and the associate Jacobians in `F{i}.jac`. Technically achieved by representing primary degrees of freedom and other quantities as types for which arithmetic operations are *overloaded* to also compute derivatives through chain rule.

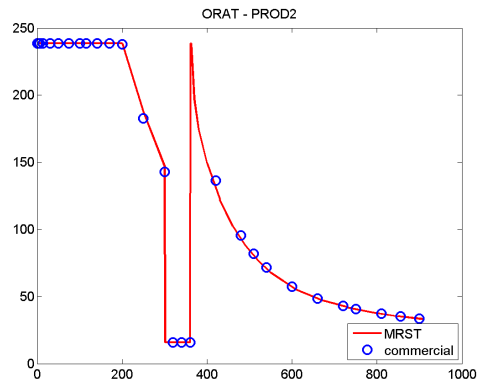
The Approach Produces Promising Results

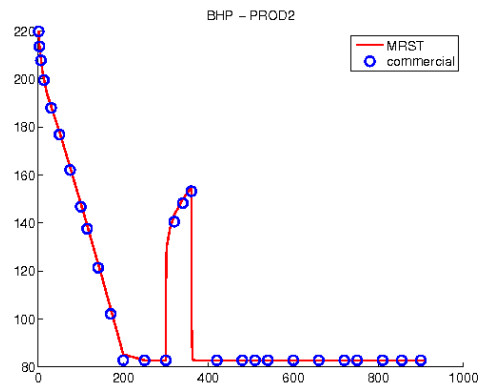
SPE 9 Benchmark Case

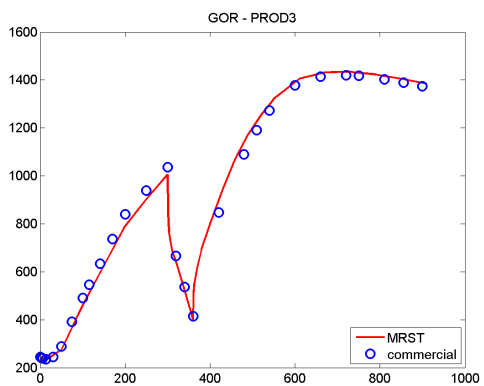
`http://www.sintef.no/Projectweb/MRST/Modules/`

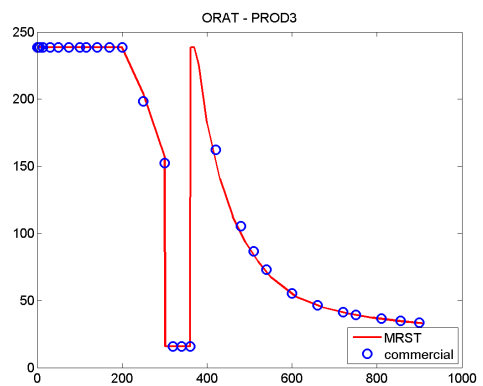
`Fully-implicit-solvers-based-on-automatic-differentiation/`

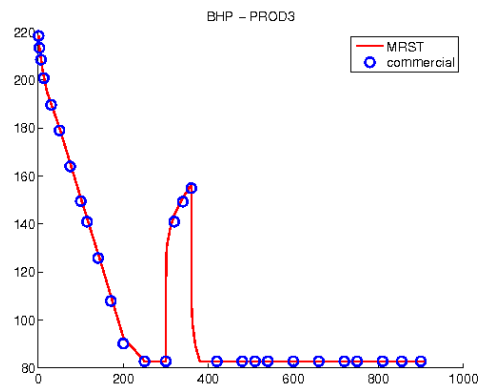












The Approach Produces Promising Results

SPE 9 Benchmark Case

`http://www.sintef.no/Projectweb/MRST/Modules/`

`Fully-implicit-solvers-based-on-automatic-differentiation/`