OPM Flow: Scaling Up an Open-Source Simulator for Industrial Application

SIAM Conference on Mathematical & Computational Issues in the Geosciences
Baton Rouge, October 16 2025

Outline

1. Motivation and introduction
2. Our well model story
3. Our automatic differentiation story
4. User programmability

What would the wells of the Troll field look like, moved to the surface?

- 2000 km of wells drilled in the reservoir
- Superimposed on the city of Bergen

Image by Hege Skyrseth (Equinor) via LinkedIn

Open Porous Media (OPM)

The Open Porous Media initiative started in 2009 aiming to:

- Connect research groups with complementary expertise
- Connect industry and researchers
- Curate open-source software and open data, in open collaboration

Main contributors to OPM:

SINTEF
NORCE
equinor
OPM-OP
ceetron solutions
TNO
CLIMIT
TotalEnergies
EuroHPC Joint Undertaking
Universität Stuttgart
BigData Accelerate

OPM Flow: An open-source reservoir simulator

Extensible and flexible

Oil/gas, CO_2 storage, geothermal, EOR, etc.

Collaborative development

Open source community centered on GitHub

Industrial relevance

Used in production by Equinor and others.

OPM Flow capabilities (highlights)

Modeling

- Black-oil model three-phase flow
- Optional thermal model
- Advanced fluid properties and models
- Multisegment well model
- *Group controls and networks*
- Industry standard input and output
- Comprehensive user manual

Methods

- Finite volume discretization with TPFA
- *Couples reservoir and well model*
- Fully implicit and coupled solution
- Newton and Nonlinear DD solvers
- *Jacobians from automatic differentiation*
- Flexible and powerful linear solvers
- MPI parallel scalable to > 1000 cores

What sets OPM Flow apart?

Uncommon for commercial software

- Open source
- Fast turnaround for research results
- Multiple providers of improvements and fixes

- Industrial compatibility
- User focus
- Handles very complex problems

Uncommon for research software

Scaling up for industry use means...

Provide support for users

Handle full asset model complexity

Robustness and performance

Integrate into industrial workflows

Control technical debt and complexity

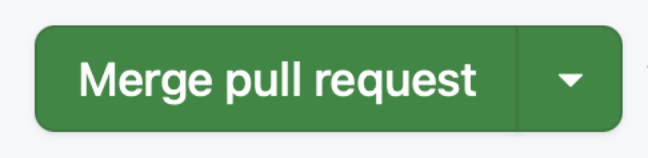
Research inside operational simulator

Integrate new research results

Developer infrastructure

Stakeholder alignment

- Stakeholders who provide funding:
  strong influence on ***what*** will be developed
- Developers and researchers have the merge button:
  decide ***how*** will it be done
- Frequent communication is key
  - formal overview meetings (30 minutes per 14 days)
  - informal task-focused meetings (typically 1-2 per week)
  - email, support tickets
- Stakeholders actively test and evaluate OPM Flow
  - See e.g. recent RSC paper:
    Reiso, E., et al. 2025. *Lessons Learned In Using Open-source Simulation Software On Real Asset Models*



Equations, extended black-oil model

This is where the well flow enters!

Mass conservation for each pseudocomponent α (water, oil, gas):

$$\frac{\partial}{\partial t}(\phi A_\alpha) + \nabla \cdot \boldsymbol{u}_\alpha + q_\alpha = 0$$

The accumulation terms A_α and fluxes $\boldsymbol{u}_\alpha$ are given by:

$$A_\alpha = m_\phi (b_\alpha s_\alpha + \sum_{\beta \neq \alpha} r_{\alpha\beta} b_\beta s_\beta)$$

$$\boldsymbol{u}_\alpha = b_\alpha \boldsymbol{v}_\alpha + \sum_{\beta \neq \alpha} r_{\alpha\beta} b_\beta \boldsymbol{v}_\beta$$

The corresponding phase velocities are given by Darcy's Law:

$$\boldsymbol{v}_\alpha = -\lambda_\alpha \boldsymbol{K}(\nabla p_\alpha - \rho_\alpha \boldsymbol{g})$$

ϕ: porosity

q_α: source term

b_α: $\rho_\alpha / \rho_{\alpha,\mathrm{ref}}$

s_α: saturation

$r_{\alpha\beta}$: solution ratio

ρ_α: density

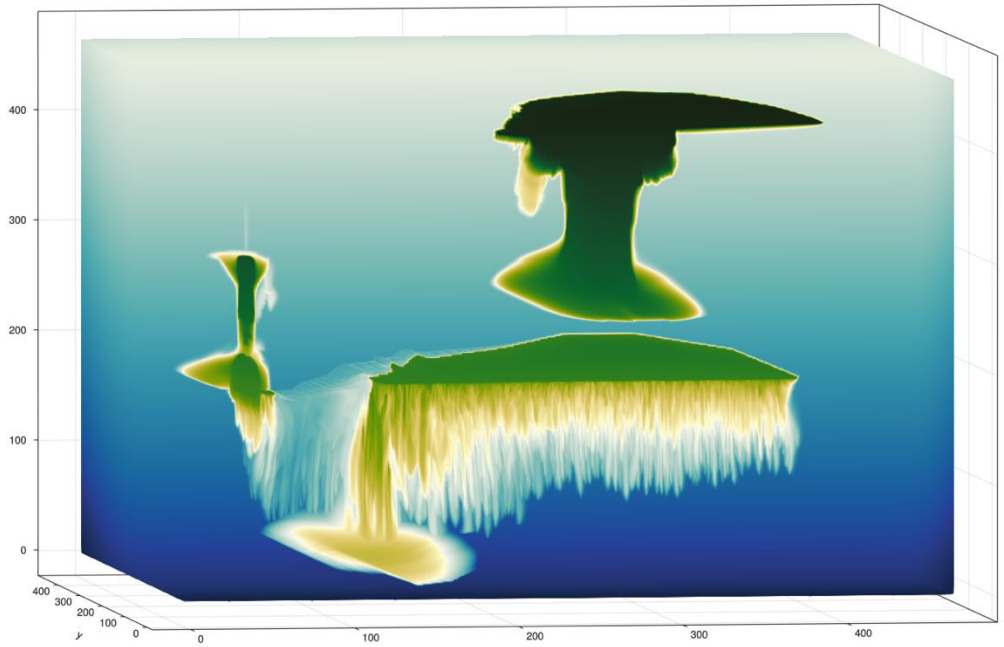
Rasmussen, A. F., Sandve, T. H.., Bao, K., et al. 2021. *The Open Porous Media Flow reservoir simulator*.

Well modeling 1: source term

Discretized conservation equation for single cell i:

$$\frac{\phi_i V_i}{\Delta t}\left(A_{\alpha,i} - A^0_{\alpha,i}\right) + \sum_{j \in C(i)} u_{\alpha,ij} + q_{\alpha,i} = 0$$

- Must define $q_{\alpha,w,i}$ – the flow of each component for each well into each cell
- Simplest model: assume these numbers are given a priori
  - Used for the SPE11 benchmark to eliminate well modeling as a factor.



Brine density for Case C from the 11th SPE Comparative Solution Project, 101-million cell simulation with OPM Flow using the CO2STORE feature

Well modeling 2: BHP and rate

Bottom Hole Pressure (BHP) model:

$$q_{\alpha,w,i} = \lambda_{\alpha,i} T_{w,i} \left(p_{w,\mathrm{bhp}} + \Delta p_{w,i} - p_i \right)$$

- Assumes as given:
  - connection factors $T_{w,i}$
  - a mobility factor $\lambda_{\alpha,i}$
  - bottom hole pressure $p_{w,\mathrm{bhp}}$
  - hydrostatic pressure adjustments $\Delta p_{w,i}$
- Gives dynamic behaviour, changing with cell pressure p_i
- Similar to a Dirichlet pressure boundary condition

Well rate model:

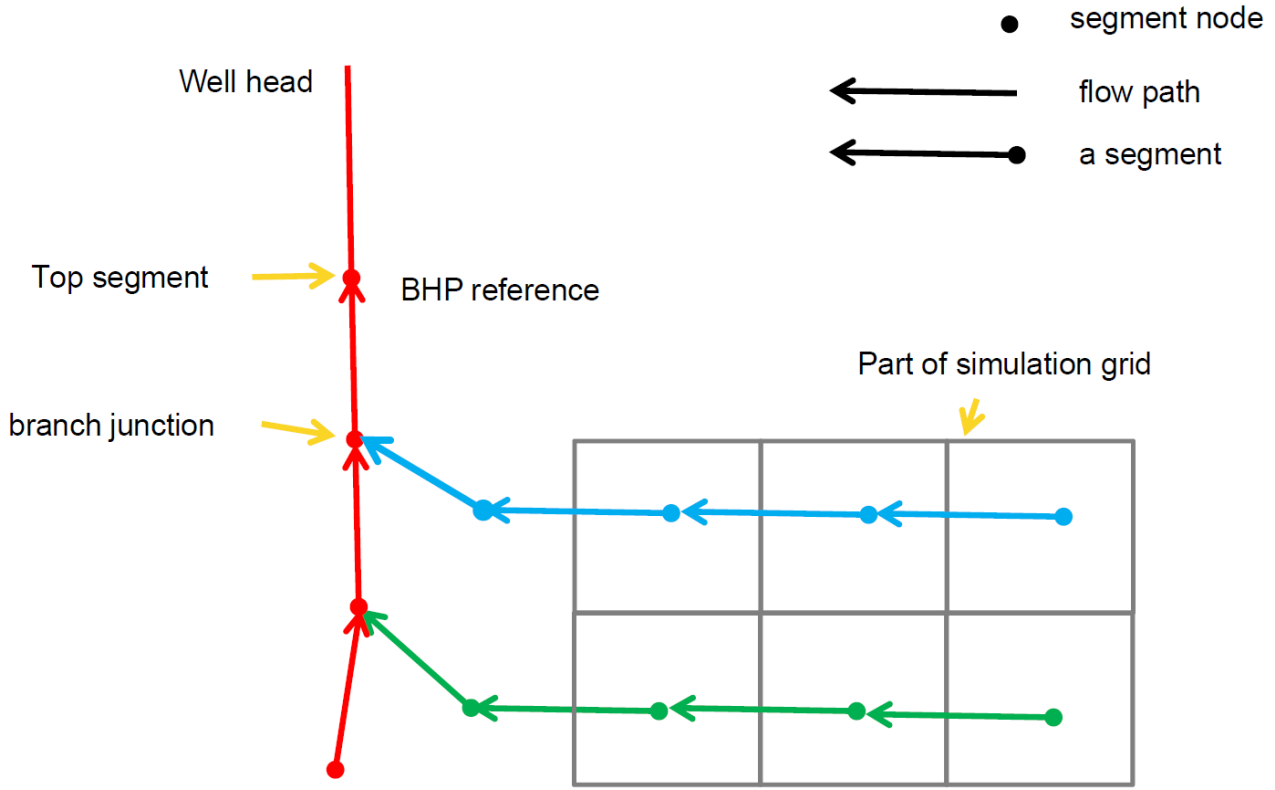
Define the total flow in a well:

$$q_{\alpha,w} = \sum_{i \in C(w)} q_{\alpha,w,i}$$

- Give a priori well rate by well ***control equation***: $q_{\alpha,w} = \overline{q_{\alpha,w}}$ (overbar denoting target value)
- This makes $p_{w,\mathrm{bhp}}$ an unknown quantity to be solved for.

Well modeling 3: x-flow and multisegment

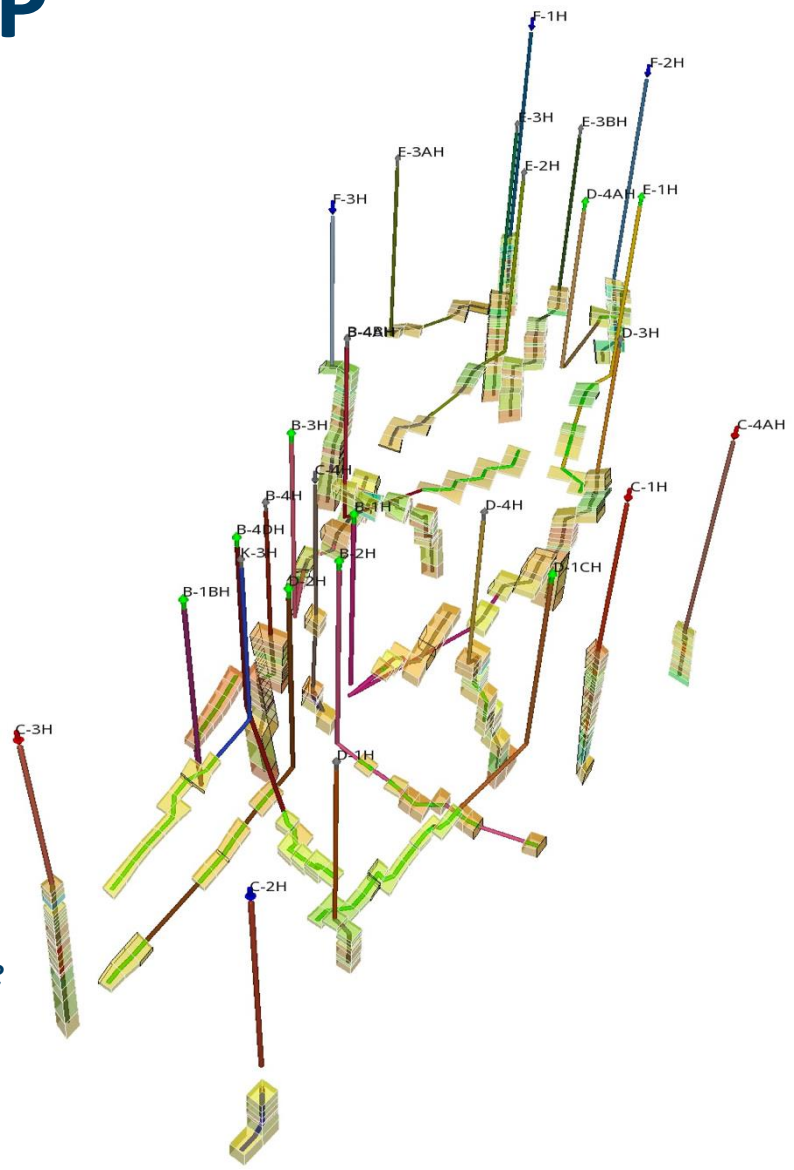
- Well rate model with fractions.
  - Add two more unknowns (water fraction and gas fraction).
  - Allows modeling cross-flow (fluid flowing both into and out of the same well).
- Multisegment well model
  - Well consists of segments, that form a tree structure.
  - Same unknowns as before, but now per segment instead of per well.
  - Equations for each segment: mass conservation + pressure
  - Advanced valve models in well



Multisegment well model, with several branches.

Well modeling 4: THP and VFP

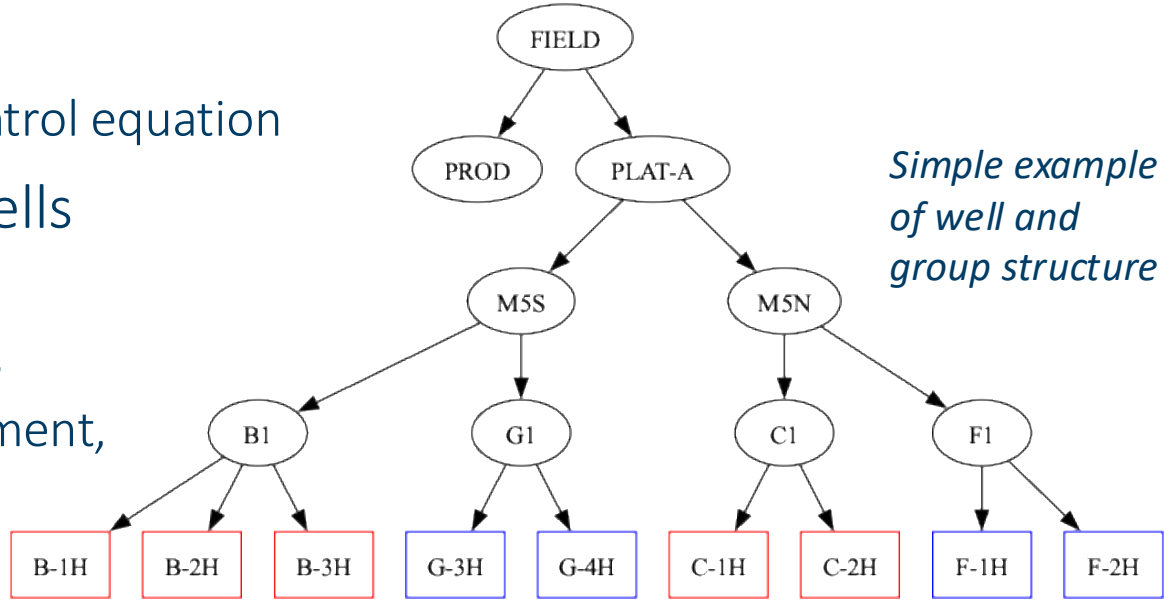
- Wells controlled by tubing head pressure (THP)
  - Assume given Vertical Flow Performance (VFP) relation for well:
    $$p_{w,\mathrm{bhp}} = f(q_{\alpha,w}, f_w, f_g, p_{w,\mathrm{thp}})$$
  - Now even more nonlinear. Models minimum rate for operability.
  - The water and gas fractions f_w, f_g complicate solution: depend on inflows $q_{\alpha,w,i}$, which depend on $p_{w,\mathrm{bhp}}$



Some of the wells of the Norne field and their connected cells

Well modeling 5: constraints and groups

- Well behaviour guided by single control equation:
  - Bhp controlled: $p_{w,\mathrm{bhp}} = \overline{p_{w,\mathrm{bhp}}}$
  - Rate controlled: $q_{\alpha,w} = \overline{q_{\alpha,w}}$
  - Thp controlled: $p_{w,\mathrm{bhp}} = f(q_{\alpha,w}, f_w, f_g, \overline{p_{w,\mathrm{thp}}})$
- Typically: multiple constraints given.
  - The strictest one at any one time becomes the control equation
- Group controls and hierarchical control of wells
  - Group rate constraints: $q_{\alpha,G} \overset{\mathrm{def}}{=} \sum_{w \in G} q_{\alpha,w} = \overline{q_{\alpha,G}}$
  - User-defined rate distribution to subordinate wells
  - New control variants: reinjection, voidage replacement, gas consumption etc.
  - How to solve/satisfy these?



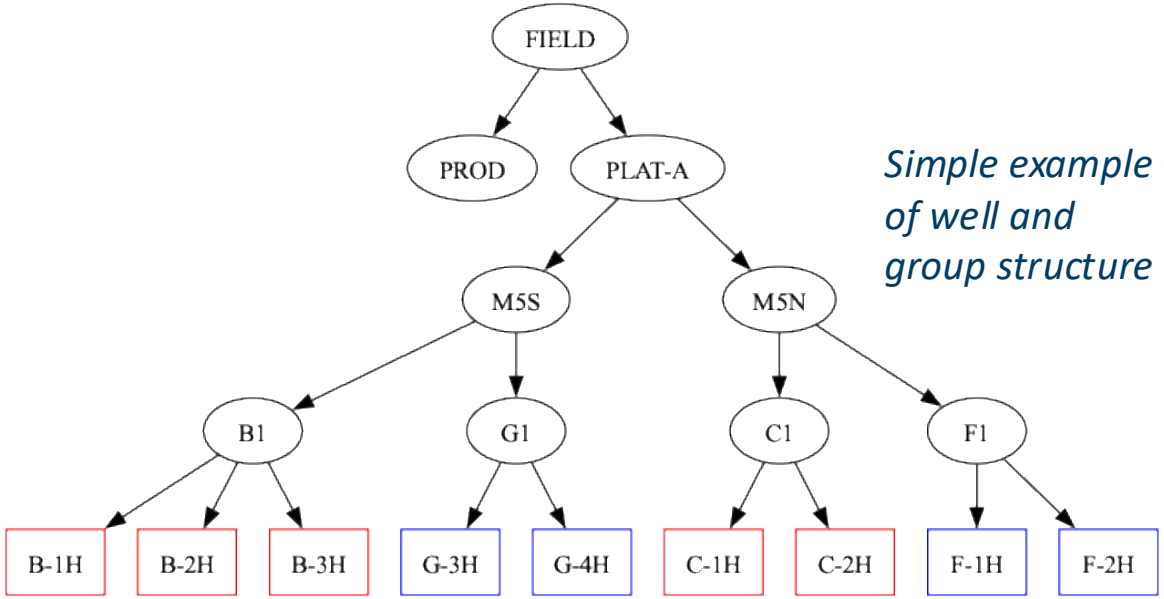
Simple example of well and group structure

Well modeling 6: network, gas lift

- Network model: dynamic THP pressure constraints on wells
  - Constraint calculated from top of tree down to wells
  - Pressure drop given by VFP tables:

$$p_{\text{child}} = f(q, f_w, f_g, p_{\text{parent}})$$

  - Simple to calculate given well flows
  - However: wells on THP control makes this implicit!
  - How to solve? Iterations outside the rest?
- Gas lift optimization
  - Allocate available gas to injection wells optimally
  - How to solve? Iterations where?



Simple example of well and group structure

Well modeling takeaways

- OPM Flow supports many advanced features well.
- Organically growing over a long time period.

Is there a catch?

- Organically grown features are not planned out in advance.
- Contains multiple ideas for how to execute.
- Process produces technical debt that must be dealt with in time.

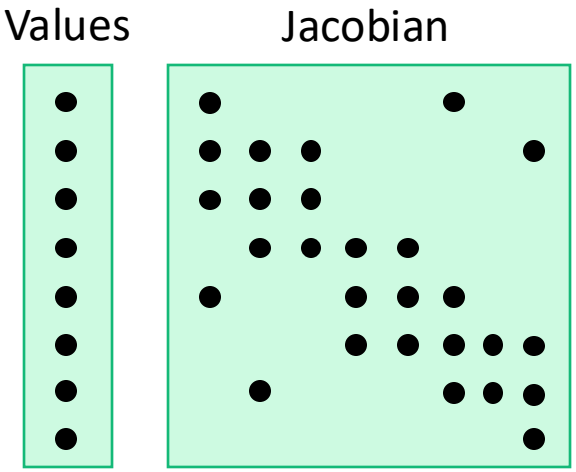
Takeaways:

- Planning is valuable – knowing where you need to go allows better initial design.
- You may end up needing a new design at some point anyway!

Automatic differentiation version 1

Original design:

- Quantities represented by a value vector and a jacobian sparse matrix
- Constants are just a value vector
- Operators as sparse matrices



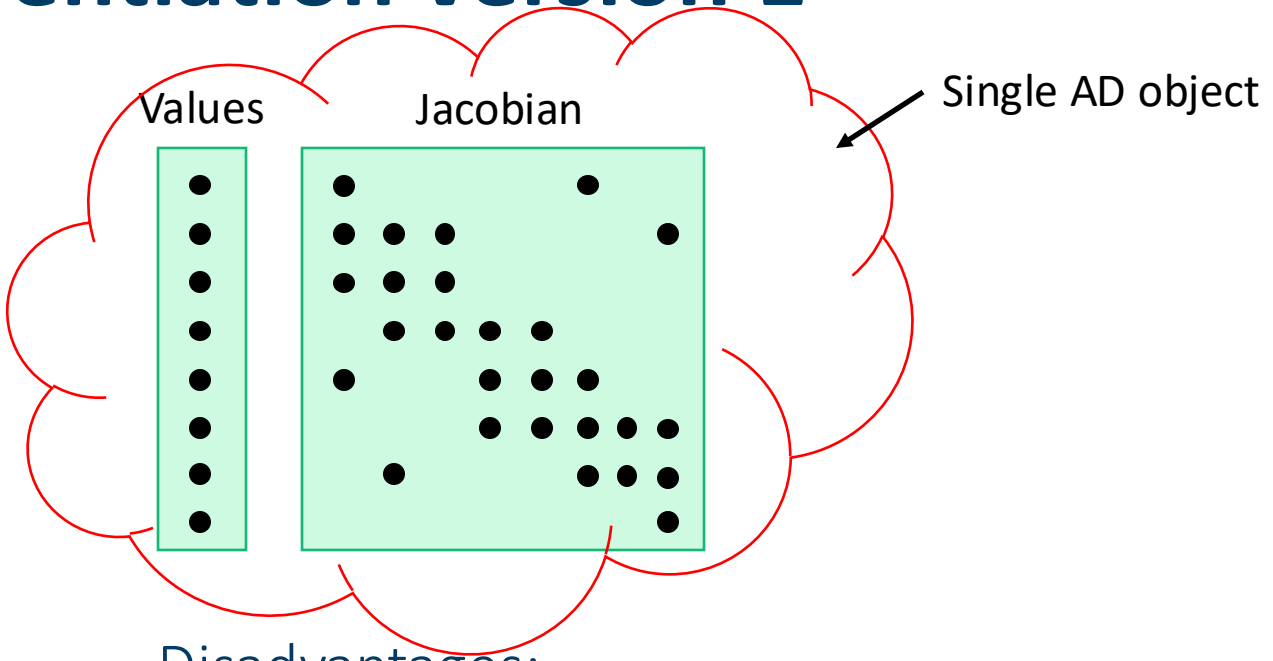

```
massflux = upwind(b * mob) * trans * dp;
material_balance = pv * dt * (accum - accum0) + ops.div * massflux;
```

Pseudocode, not actual code!

Automatic differentiation version 1

Original design:

- Quantities represented by a value vector and a jacobian sparse matrix
- Constants are just a value vector
- Operators as sparse matrices



Advantages:

- Notation similar to math
- Automatically get both sparsity structure and derivative values

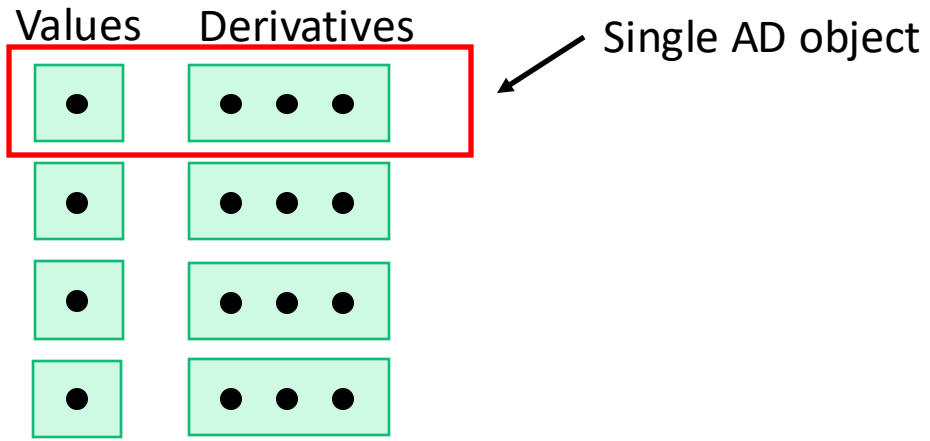
Disadvantages:

- Slow
  - creating new sparse matrices every operation
  - fusing operations is hard
- Tricks to improve performance increased complexity

Automatic differentiation version 2

New design:

- Quantities represented by a single scalar and a fixed-size small vector of derivatives
- Constants are just a scalar
- No operator abstraction, put blocks into sparse matrix manually



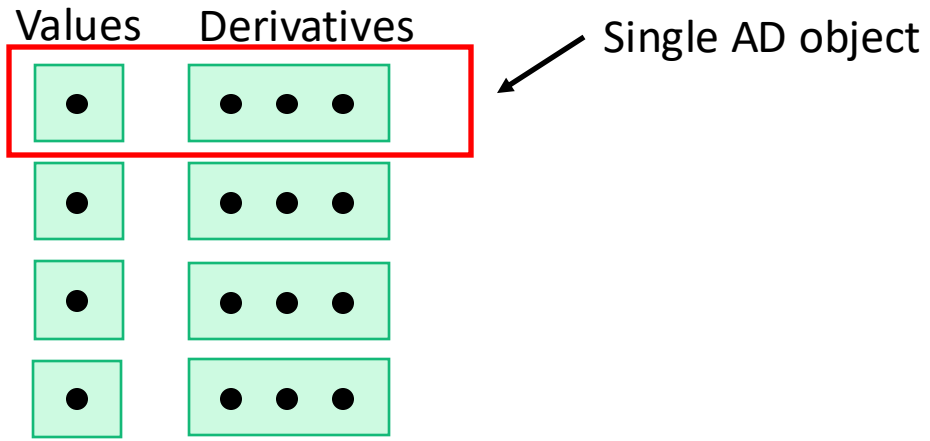

```
mat_balance[i] = pv * dt * (accum[i] - accum0[i]);
for (j : C(i)) {
    massflux[i, j] = upwind(b[i, j] * mob[i, j]) * trans[i, j] * (p[i] - p[j]);
    mat_balance[i] += massflux[i, j];
    jacobian[j, i] = -massflux[i, j].derivative();
}
residual[i] = mat_balance[i].value();
```

Pseudocode, not actual code!

Automatic differentiation version 2

New design:

- Quantities represented by a single scalar and a fixed-size small vector of derivatives
- Constants are just a scalar
- No operator abstraction, put blocks into sparse matrix manually



Advantages:

- Much faster
- AD class is much simpler

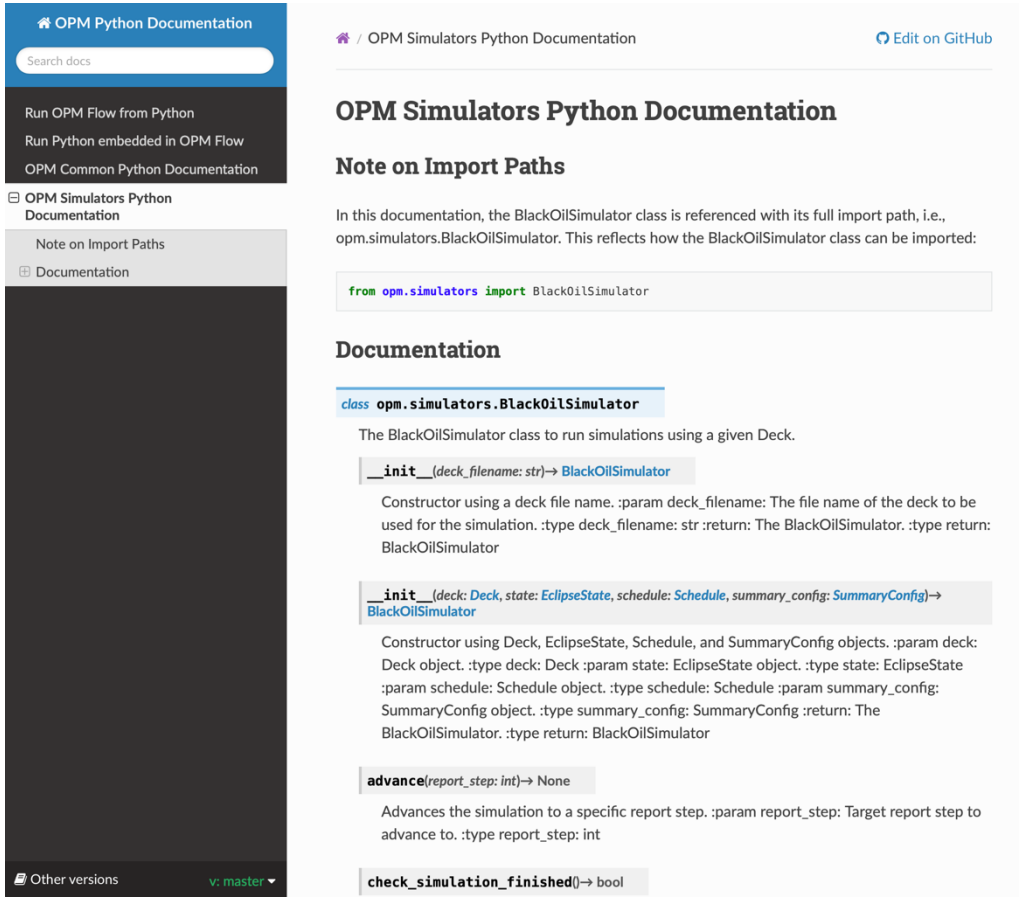
Although not without cost:

- New approach was developed separately
- Making the change merged two different codes with different style and approach
- Resulting mixed code was functional, but gained technical debt

Lauser, A., Rasmussen, A. F., Sandve, T. H., et al. 2018. *Local forward-mode automatic differentiation for high performance parallel pilot-level reservoir simulation*

User-controlled programmability

- ACTIONX
  - Allows user variables and conditionals in the simulation deck to trigger wide range of actions
- PYACTION
  - Extends the above to run Python code that can trigger actions or modify the simulator state
- Python interface
  - Can run the simulator or parts of it as a Python script



User-controlled programmability

Python interface: lets users run Flow as a Python script

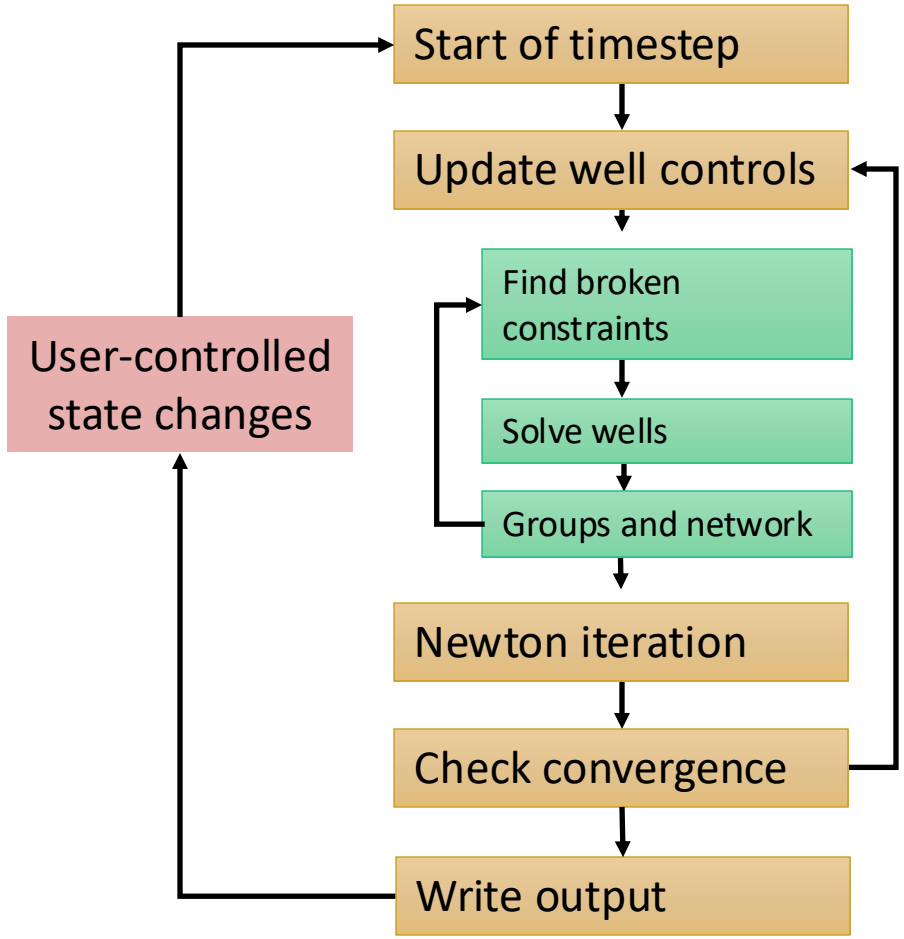
Example from Lisa Julia Nebel and Håkon Hægland

```
import os
import sys
from opm.simulators import BlackOilSimulator
from opm.io.parser import Parser
from opm.io.ecl_state import EclipseState
from opm.io.schedule import Schedule
from opm.io.summary import SummaryConfig

deck   = Parser().parse('SAMPLE_DECK.DATA')
state  = EclipseState(deck)
schedule = Schedule( deck, state )
summary_config = SummaryConfig(deck, state, schedule)

sim = BlackOilSimulator(deck, state, schedule, summary_config)
sim.step_init()
sim.step()
poro = sim.get_porosity()
poro = poro *.95
sim.set_porosity(poro)
sim.step()
sim.step()
sim.step()
sim.step_cleanup()
```

User-controlled programmability

- Forces assumptions to change:
  - well behaviour not fully determined by input
  - cannot know at the start of simulation which wells will open and when etc.
  - state can be modified
  - must change program flow to adapt to new possibilities

Simplified program flow:



Conclusions

- User and stakeholder involvement is key.
- It is hard to maintain both a high pace of new features and a code that is simple.
- Long-term planning will help choosing good designs.
  - Must still be prepared to reconsider.
- Flexible code is important, flexible developers even more so!
- Having multiple alternatives (MRST, Jutul etc.) makes it easier to get new methods tested and implemented.

SINTEF
75 years

Thank you for your attention!