SINTEF

OPM FLOW IN MSO4SC

AND OTHER STORIES

Atgeirr Flø Rasmussen

About SINTEF

Vision: **technology for a better society**

- independent, not-for-profit organization
- largest for-contract research in Scandinavia, fourth largest in Europe
- 2100 employees
- NOK 3.1 billion turnover, 90% ’won’ in open competition
- more than 7000 research projects for some 2300 clients
- offices in Trondheim, Oslo, Bergen, Brussels, Houston, . . .

Computational Geosciences group

- One of eight research groups at the department of Mathematics & Cybernetics, SINTEF Digital
- Eleven researchers/postdocs/PhD students
- Offices in Oslo, Norway
- Performs a mixture of basic and applied research
- Well known for our *open-source software*: MRST and OPM
- Internationally oriented
- Strong publication record
- Main clients: Statoil, ExxonMobil, Research Council of Norway, Wintershall, . . .

Some stories of mathematical software

SINTEF

Some stories of mathematical software

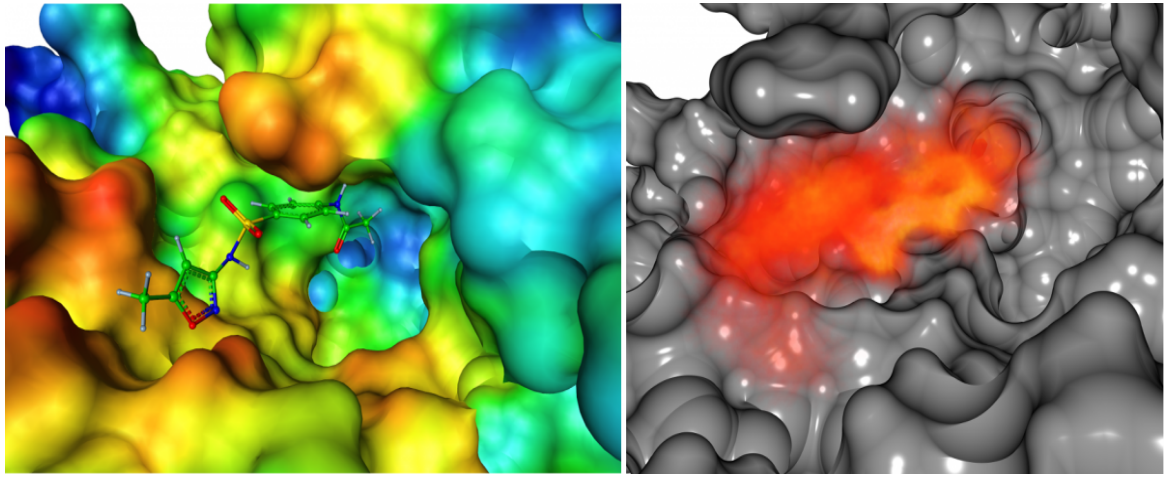
- Modeling, simulation, optimization for societal challenges (MSO4SC)
- The Open Porous Media initiative
- Flow: from proof-of-concept to deployable simulator
- OPM Flow in MSO4SC
- A problem with grid interfaces
- Collaboration joys and pains
- Making Flow perform well

Modeling, simulation, optimization for societal challenges (MSO4SC)

Motivation

Societal challenges

- … in health, energy, climate, infrastructure, pollution,
- … benefit from matematical modeling, simulation and optimization
- … are highly complex problems



Expertise required

- … in the problem domain,
- … in numerical and other mathematics
- ... in programming, parallelization, HPC
- … in databases and visualization

Typically not easily available to decision makers!

The MSO4SC project

H2020 project started in late 2016

Main ideas:

- Provide mathematical technology as a service
- … through an HPC oriented cloud e-infrastructure
- Lower the barrier to using MSO software!

What we do

- Build an online portal and repository for MSO software
- Make it simple and quick to run MSO software
- Run and scale on cloud or HPC facilities

Partners:

- ATOS (Spain)
- TU Berlin (Germany)
- Uni. Strasbourg (France)
- SINTEF (Norway)
- BCAM (Spain)
- Szechenyi Istvan Uni. (Hungary)
- Konrad-Zuse Zentrum (Germany)
- CESGA (Spain)
- KTH (Sweden)
- EU-MATHS-IN (Netherlands)

SINTEF

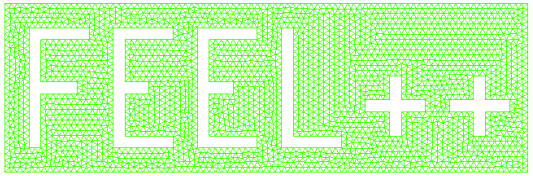
Mathematical frameworks (all open source)

FEniCS and FEniCS-HPC

- Automated solution of PDEs
- Finite element methods, weak form
- Strong parallel scalability

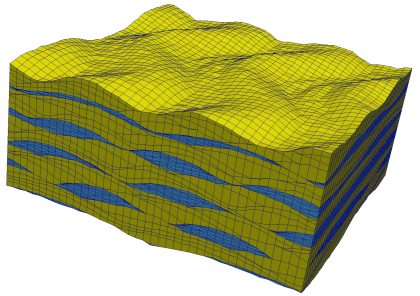
Feel++

- Embedded Domain-Specific Language (DSL) in C++
- Galerkin methods
- Shield user from solver/parallel complexity



Open Porous Media (OPM)

- Collection of C++ components and programs
- Finite volume methods
- Focus on industrial usage

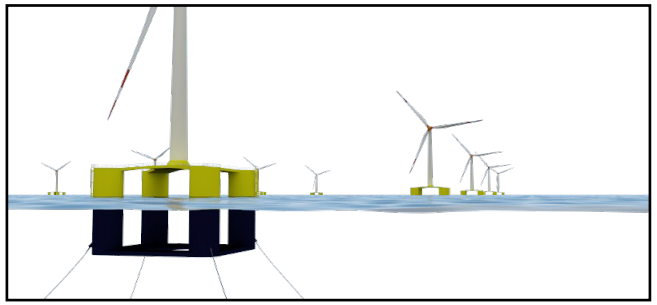


SINTEF

Pilot applications (I)

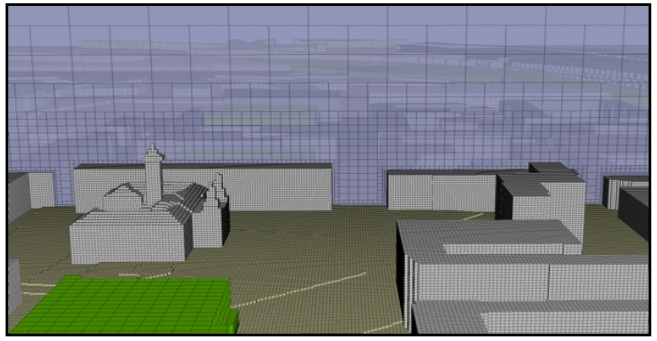
FloatingWindTurbine (BCAM)

- Fluid-structure interaction



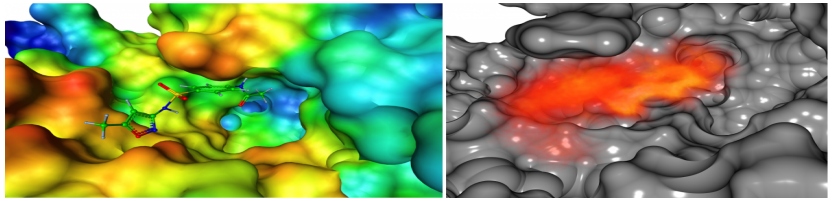
3DAirQualityPrediction (SZE, KTH)

- CFD, integration of real-time data



ZIBAffinity (ZIB)

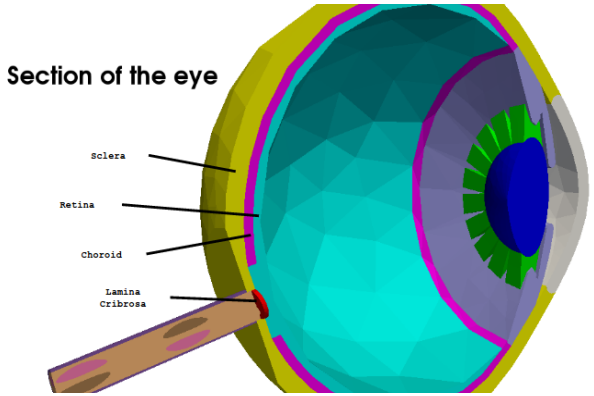
- Molecular affinity and binding energy



Pilot applications (II)

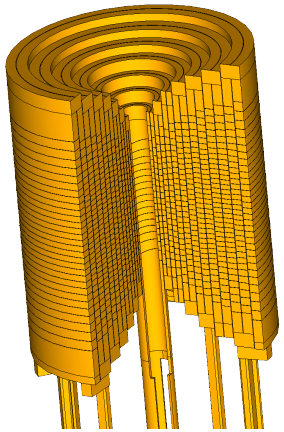
Eye2Brain (UNISTRA)

- Biological system simulation



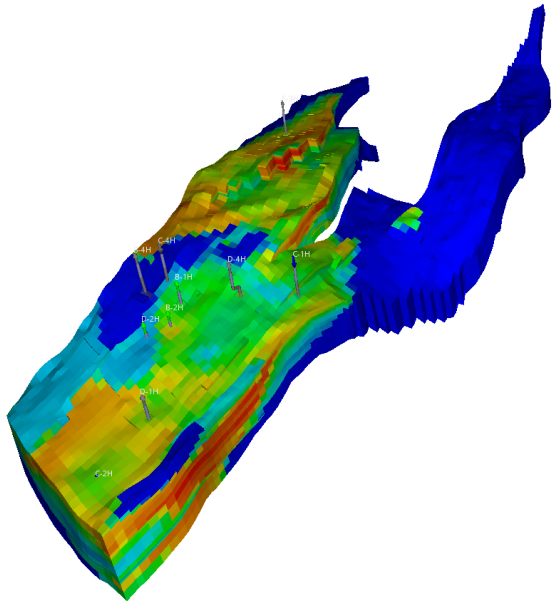
HifiMagnet (UNISTRA)

- Coupled nonlinear el-mag. to 35 T



OPM Flow (SINTEF)

- Multiphase flow in porous medium



SINTEF

The MSO4SC Portal

Will offer MSO software with

- No installation
- Easy scaling on cloud/HPC

Catalog of software

- MSO frameworks
- MSO applications
- Extensible

Data catalog (ckan)

- Open benchmark cases
- Sharing and learning opportunities

The Open Porous Media initiative

SINTEF

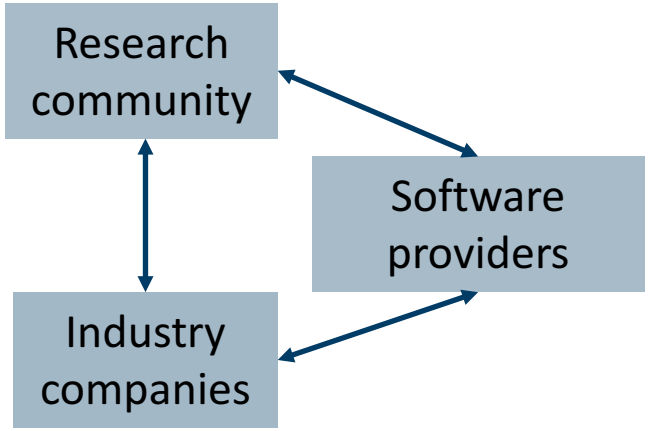
The Open Porous Media initiative

- Open Porous Media software components are or have been developed by:
  - Companies (Statoil, Total)
  - Research institutes (SINTEF, IRIS, TNO)
  - Universities (U. Stuttgart, NTNU)
  - Consultants
- Financing from industry and public (RCN, EU)
- **Open source allows easier collaboration!**

The Open Porous Media initiative – origin

Started in 2009 to combine strengths:

- Grids and discretizations (SINTEF)
- Advanced fluid models (U. Stuttgart, U. Bergen)
- Industrial know-how and funding (Statoil)
- Build on the DUNE project (many contributors)

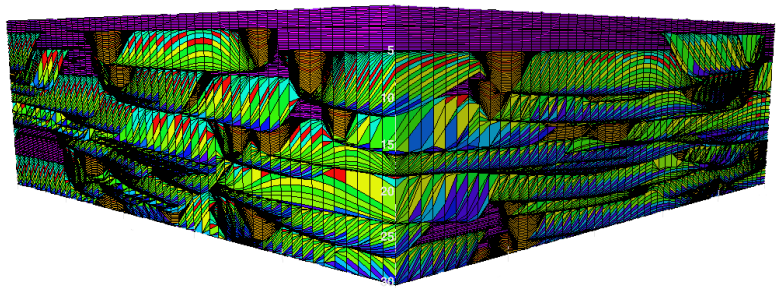


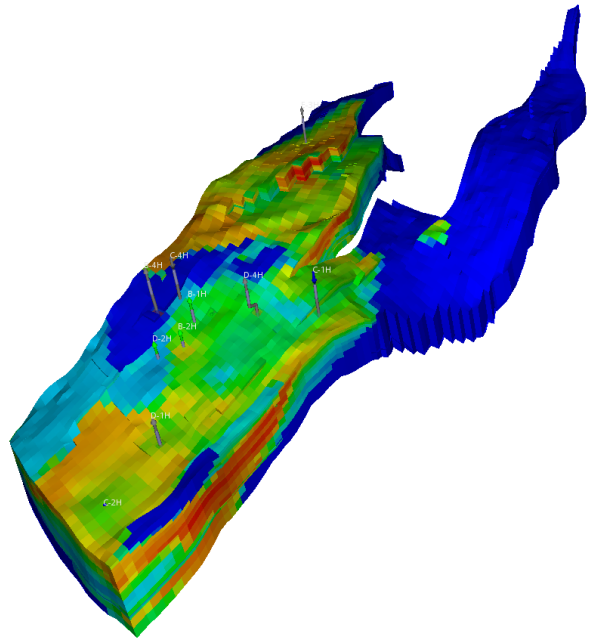
Vision: a long-lasting, efficient, and well-maintained, open-source software for flow and transport in porous media

Ambition: to be a strong base for both industrial development and academic research

What makes reservoir problems hard?

- Porous medium is strongly heterogeneous and anisotropic.
- Grids with high aspect ratio, fully unstructured, polygonal cells.
- Nontrivial phase behaviour. Phases can appear and disappear as fluid components dissolve or vaporize.
- Coupling to wells can connect regions that are far away from each other.
- The models are highly nonlinear.





The OPM initiative, 2009-2013

Collaboration with U. Stuttgart

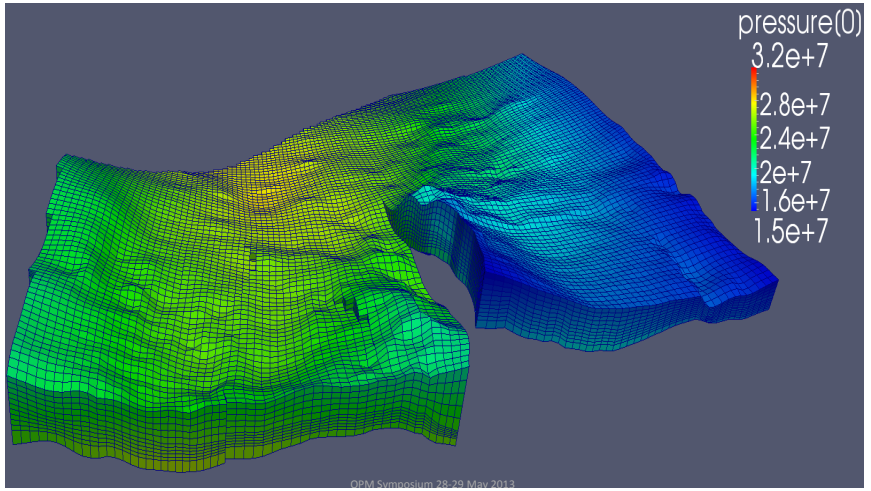
- Solving various fluid problems (Stuttgart) on corner-point grids using the CpGrid class (SINTEF)

Innovative simulator for polymer-EOR

- Reordering nonlinear solvers, improved stability

Joint Industry Project with SINTEF, IRIS, Statoil and Total

- Aim: build framework for proof-of-concept and prototype simulators
- Builds IMPES-type simulators for black-oil and CO2-injection problems
- Towards the end of the project: fully implicit black-oil simulator based on AD (what would become today's Flow)



SINTEF

The OPM initiative, 2013

A transformative year!

Fully-implicit black-oil simulator gets attention of industrial partner

- Becomes main target for development (eventually receives the name Flow)
- In retrospect: reduced focus on numerics, increased focus on industrial usability

New projects fund development

- Direct funding from industry
- Funding from Climit to make simulator usable for CO2-EOR and CO2-storage studies

Close collaboration between SINTEF, IRIS, Statoil and some German contributors

The OPM initiative, 2014-2017

Main focus still on Flow and industrial usability

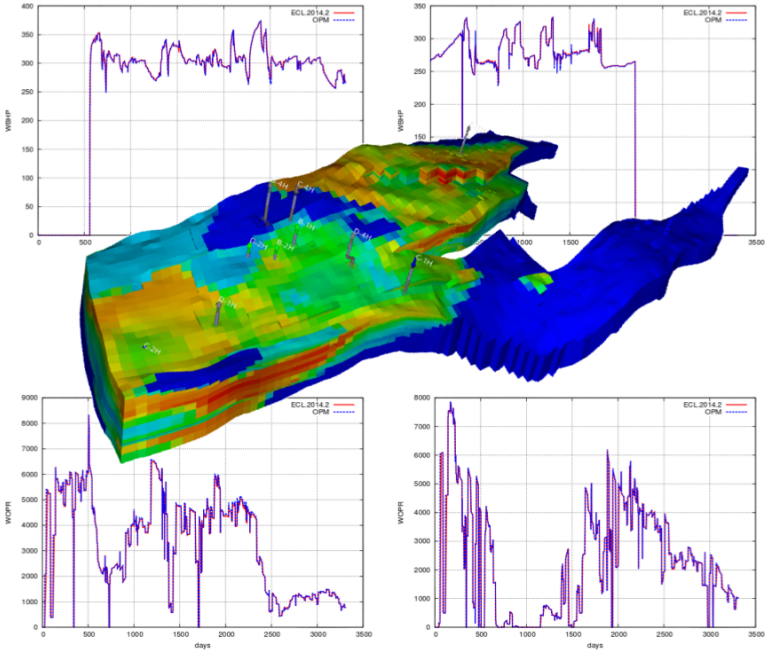
- Robustness
- Performance
- Eclipse-compatible input *and* output
- Well and group controls, multi-segment wells,
- Including solvent model (for CO2 uses) and polymer model
- MPI parallelism, exploiting the parallel Dune capabilities (moderately)

Collaboration still strong among SINTEF, IRIS and Statoil etc.

- University of Stuttgart not really involved with Flow, but using other parts (grid etc.)

New groups are interested

- TNO is now participating, new academic and industrial groups joining



SINTEF

Flow: from proof-of-concept to deployable simulator

Before Flow

Stein Krogstad introduces *automatic differentiation* (AD) to the Matlab Reservoir Simulation Toolbox (MRST)

"*a set of techniques to numerically evaluate the derivative of a function specified by a computer program. AD exploits the fact that every computer program, no matter how complicated, executes a sequence of elementary arithmetic operations (addition, subtraction, multiplication, division, etc.) and elementary functions (exp, log, sin, cos, etc.). By applying the chain rule repeatedly to these operations, derivatives of arbitrary order can be computed automatically, accurately to working precision, and using at most a small constant factor more arithmetic operations than the original program.*"

Creates (our) first fully implicit black-oil simulator using AD

AD techniques already used in GPRS-AD, others

```
% ----------------------------------------------------------------
[krW, krO, krG] = f.relPerm(sW, sG);

% water props (calculated at oil pressure OK?)
bW     = f.bW(p);
%bW     = f.bW(p-pcOW);
rhoW   = bW.*f.rhoWS;
% rhoW on face, avarge of neighboring cells (E100, not E300)
rhoWf  = s.faceAvg(rhoW);
mobW   = trMult.*krW./f.muW(p);
%mobW   = trMult.*krW./f.muW(p-pcOW);
dpW     = s.grad(p-pcOW) - g*(rhoWf.*s.grad(G.cells.centroids(:,3)))
% water upstream-index
upc = (double(dpW)>=0);
bWvW = s.faceUpstr(upc, bW.*mobW).*s.T.*dpW;


% oil props
bO     = f.bO(p, rs, isSat);
rhoO   = bO.*(rs*f.rhoGS + f.rhoOS);
rhoOf  = s.faceAvg(rhoO);
mobO   = trMult.*krO./f.muO(p,rs,isSat);
dpO    = s.grad(p) - g*(rhoOf.*s.grad(G.cells.centroids(:,3)));
% oil upstream-index
upc = (double(dpO)>=0);
bOvO   = s.faceUpstr(upc, bO.*mobO).*s.T.*dpO;
rsbOvO = s.faceUpstr(upc, rs).*bOvO;

% gas props (calculated at oil pressure OK?)
bG     = f.bG(p);
%bG     = f.bG(p+pcOG);
rhoG   = bG.*f.rhoGS;
rhoGf  = s.faceAvg(rhoG);
mobG = trMult.*krG./f.muG(p);
%mobG = trMult.*krG./f.muG(p+pcOG);

dpG     = s.grad(p+pcOG) - g*(rhoGf.*s.grad(G.cells.centroids(:,3)))
% water upstream-index
upc = (double(dpG)>=0);
bGvG = s.faceUpstr(upc,bG.*mobG).*s.T.*dpG;



% EQUATIONS ------------------------------------------------------
isSat0 = (double(sG0)>0);
rsSat  = f.rsSat(p);


% oil:
eqs{1} = (s.pv/dt).*( pvMult.*bO.*(1-sW-sG) - pvMult0.*f.bO(p0,rs0,
```

”Flow” in 2013

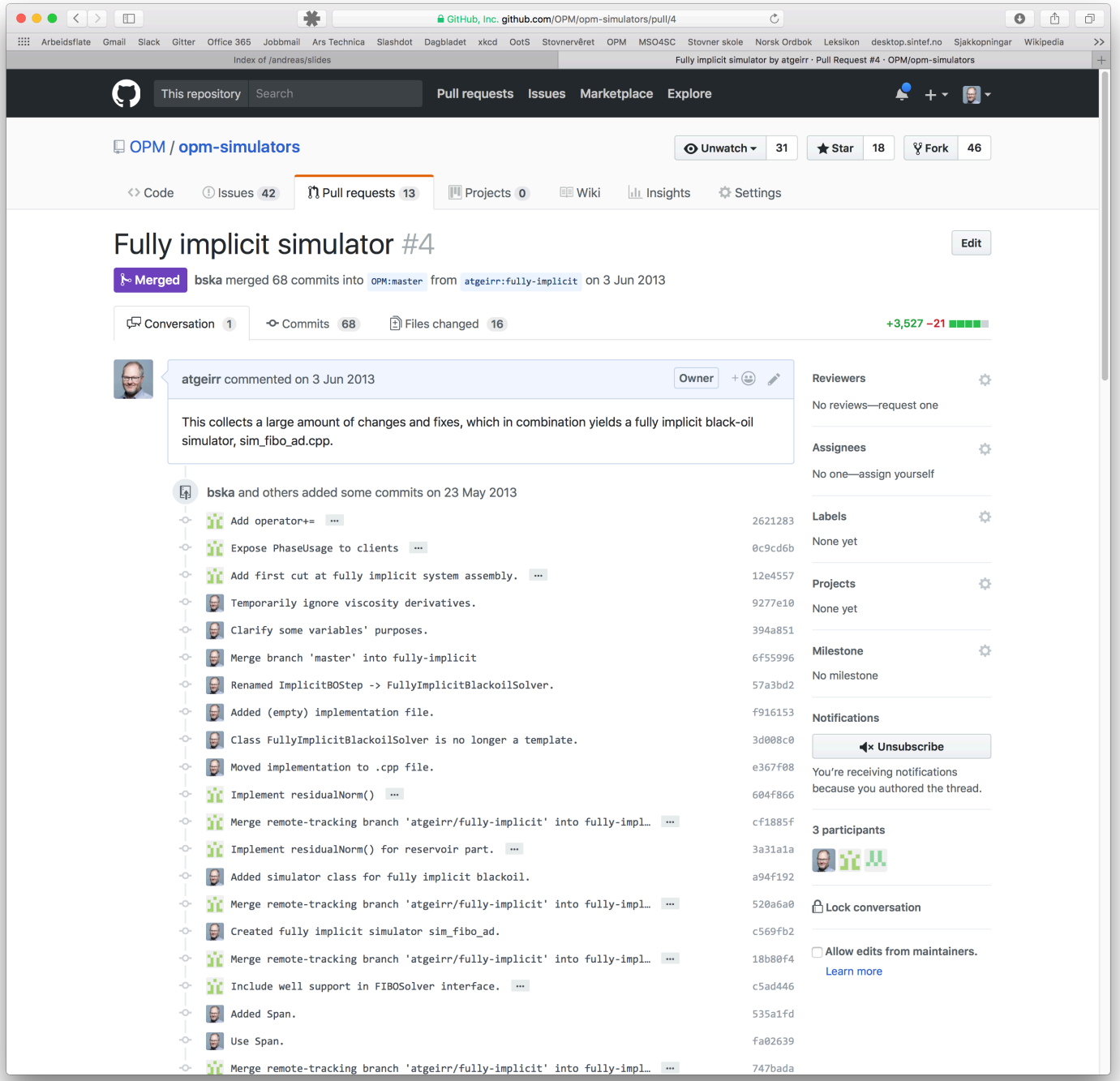
Name: ”sim_fibo_ad” (very catchy!)

Able to run SPE1 (only very simple and small test cases)

Written using small AD library similar to MRST’s AD class (1 week development)

Originally: wanted to use GPRS’ library

1 month later: first version done (as well as 2p pressure/transport/impes solvers)



What is all the fuss about?

Prototype gets attention from industrial partner (more than expected)

Industrial partner convinces SINTEF and IRIS to focus C++ development on fully implicit simulator

Some reasons:

- Fully implicit method is the industrial standard
  - Research results will be measured against this
- Impatient with commercial vendors
  - Commercial software cycle slow
- Eager to make research groups to work together

```
    for (int phaseIdx = 0; phaseIdx < fluid_.numPhases(); ++phaseIdx)
        const std::vector<PhasePresence>& cond = phaseCondition();
        sd_.rq[phaseIdx].mu = asImpl().fluidViscosity(canph_[phaseIdx
                                                      state.canonical
                                                      state.temperatu
        sd_.rq[phaseIdx].rho = asImpl().fluidDensity(canph_[phaseIdx]
        asImpl().computeMassFlux(phaseIdx, trans_all, sd_.rq[phaseIdx
                                 state.canonical_phase_pressures[canp

        residual_.material_balance_eq[ phaseIdx ] =
            pvdt_ * (sd_.rq[phaseIdx].accum[1] - sd_.rq[phaseIdx].acc
            + ops_.div*sd_.rq[phaseIdx].mflux;
    }

    // -------- Extra (optional) rs and rv contributions to the mass

    // Add the extra (flux) terms to the mass balance equations
    // From gas dissolved in the oil phase (rs) and oil vaporized in
    // The extra terms in the accumulation part of the equation are a
    if (active_[ Oil ] && active_[ Gas ]) {
        const int po = fluid_.phaseUsage().phase_pos[ Oil ];
        const int pg = fluid_.phaseUsage().phase_pos[ Gas ];

        const UpwindSelector<double> upwindOil(grid_, ops_,
                                               sd_.rq[po].dh.value());
        const ADB rs_face = upwindOil.select(state.rs);

        const UpwindSelector<double> upwindGas(grid_, ops_,
                                               sd_.rq[pg].dh.value());
        const ADB rv_face = upwindGas.select(state.rv);

        residual_.material_balance_eq[ pg ] += ops_.div * (rs_face *
        residual_.material_balance_eq[ po ] += ops_.div * (rv_face *

        // OPM_AD_DUMP(residual_.material_balance_eq[ Gas ]);

    }


    if (param_.update_equations_scaling_) {
        asImpl().updateEquationsScaling();
    }

}
```

”Flow” in 2014

New input deck reader: opm-parser (by Statoil)

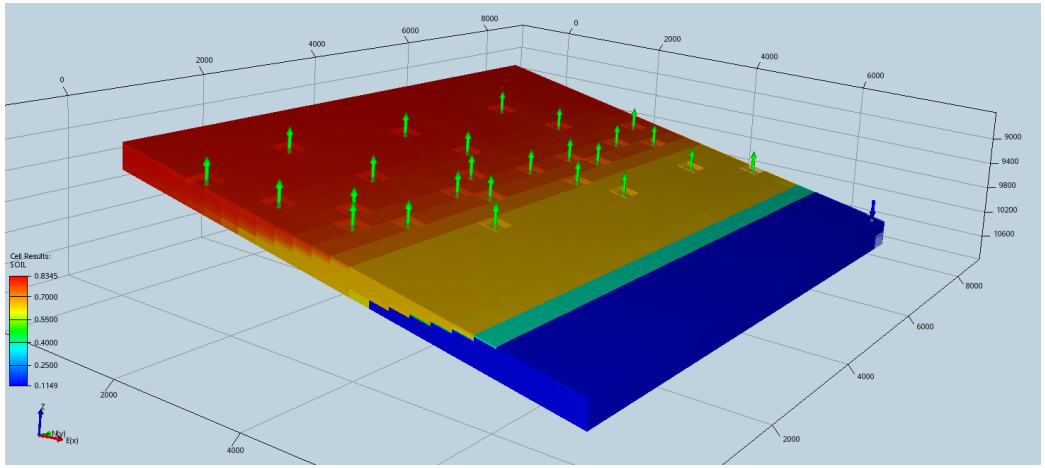
- Allows high degree of Eclipse compatibility
- Manipulate state variables like PORV, transmissibilities
- Supports SCHEDULE section well

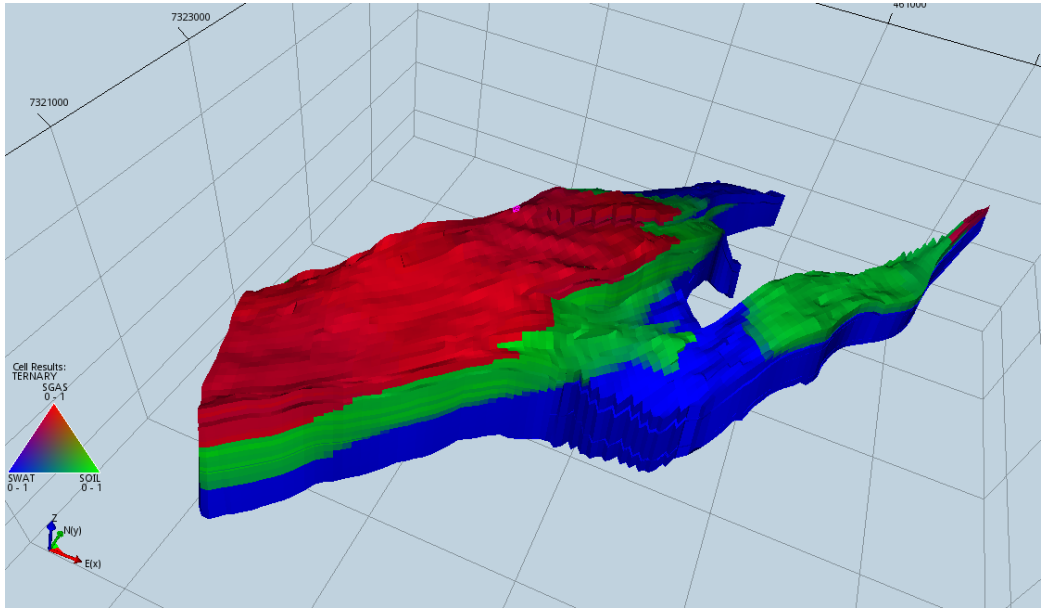
Able to run SPE9 (spring) and Norne (fall)

- Mostly matching Eclipse results
- This was a huge effort, implementing dozens of features small and large in order to match
- Bad performance: SPE9 takes 3 minutes…

No OPM release this year

- Concentrating on improving Flow
- In retrospect, not a good idea





SINTEF

Flow in 2015

MPI-parallel version working

- Poor scaling, not fully feature-complete

New name for simulator: Flow

Black-oil + polymer EOR, black-oil + solvent (CO2)

Improved Eclipse match

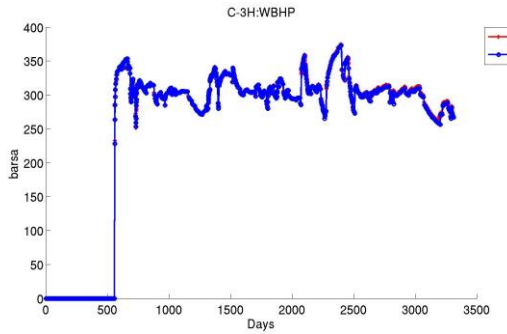
- Dozens more small fixes and features

Performance improvements

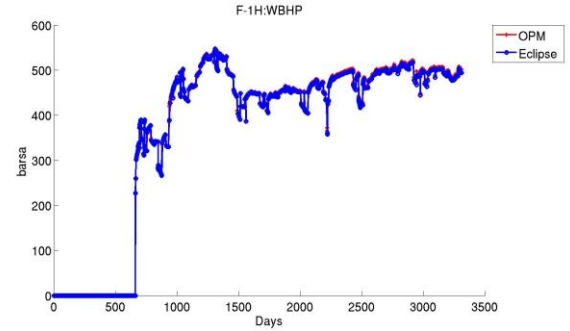
- ~6 times slower than Eclipse on Norne in March 2015
- ~3 times slower in October 2015

BHP

C-3H



F-1H



SINTEF

Flow in 2016

Multi-segment wells

- Initially not completely integrated with other features
- Only handling a subset of Eclipse features

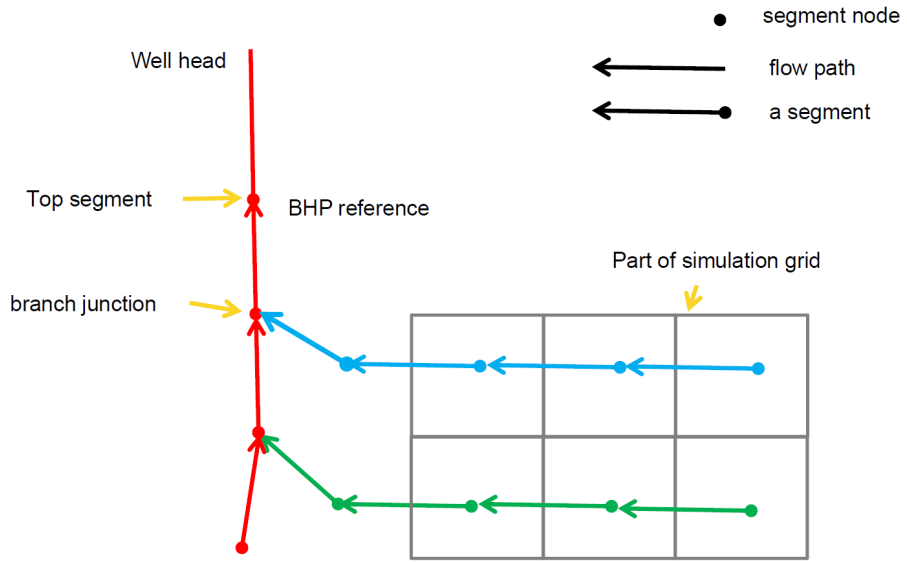
(both points much improved in 2017)

New output facilities

- Summary and restart output much improved
- Configurable from deck
- Completely new log-type output facility (to terminal and PRT files)

Performance improvements

- ~1.7 times slower than Eclipse on Norne in October 2016
- MPI parallel version scales on workstations (not HPC-level)



SINTEF

Flow in 2017

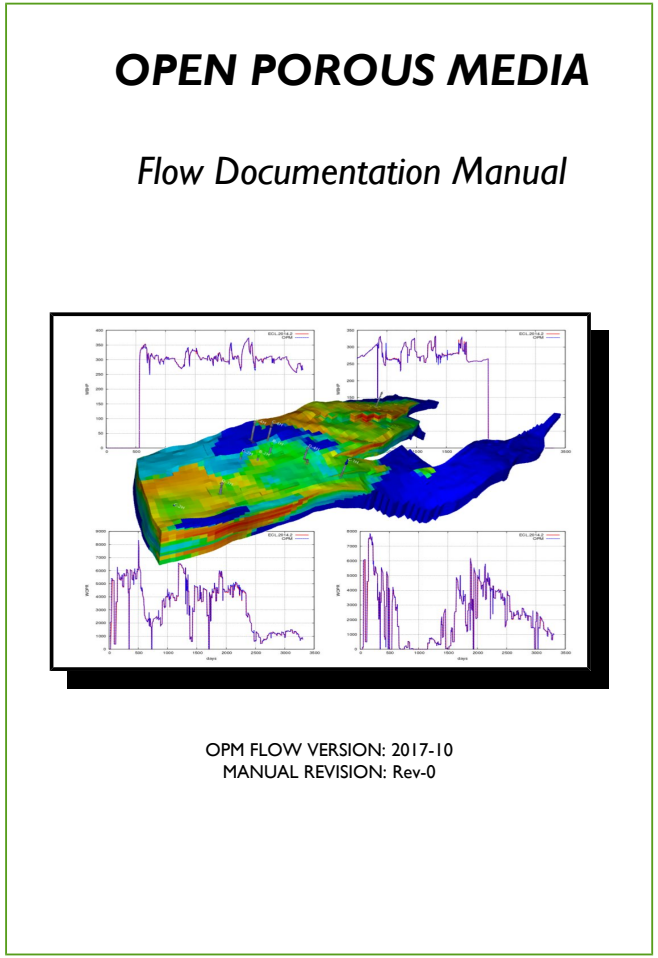
Performance improvements

- ~1.1 times slower than Eclipse on Norne in October 2017
- ~0.9 times slower than Eclipse (so: faster!) on another real reservoir
- 5 times faster than Eclipse for some solvent/CO2 models!
- Robustness: equal to or better than Eclipse on target ensembles

Manual released in October

Containerization: Docker, Singularity

- EU-project *MSO4SC* supports cloud/HPC effort
- Flow runs on any platform through containers



SINTEF

What are we* currently working on

* not just SINTEF

Performance and ease of use

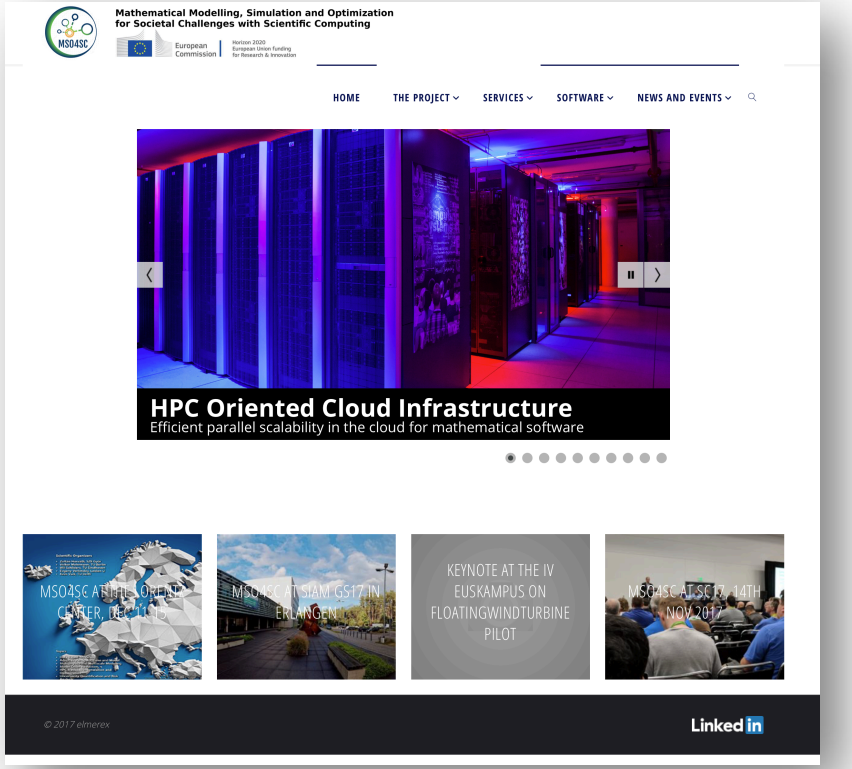
- Linear solver improvements, better preconditioning
- Better parallel scalability
- Run Flow in cloud/HPC with ”one click” deployment

New features

- Adjoints
- Thermal option
- More and improved CO2 fluid models
- Features needed to run on more field models (such as aquifers)

New methods

- Sequential implicit methods, reordering
- Higher-order methods



SINTEF

Where would we like to go with Flow (I)?

Realize ambition: *to be a strong base for both industrial development and academic research*

- Strong international collaboration
- Be the default companion to MRST in research and education
- Address industry needs

Continuous improvements to existing code base

- performance and scaling
- ease of use and deployment
- features (dual-porosity, aquifers etc.)
- methods (consistent discretizations, higher order etc.)
- robustness
- flexibility
- ease of programming

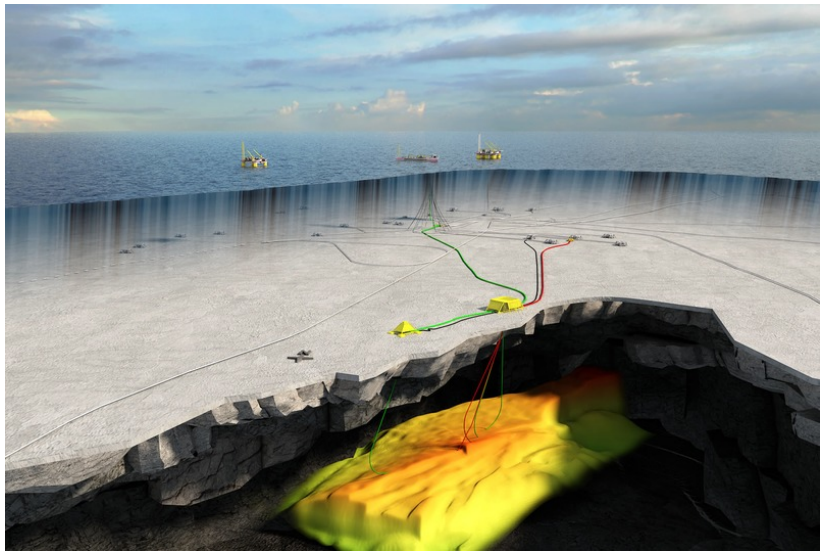


Photo credit: Statoil

Where would we like to go with Flow (II)?

Covering future needs

- Compositional models?
- Fracture flow?
- Huge models?

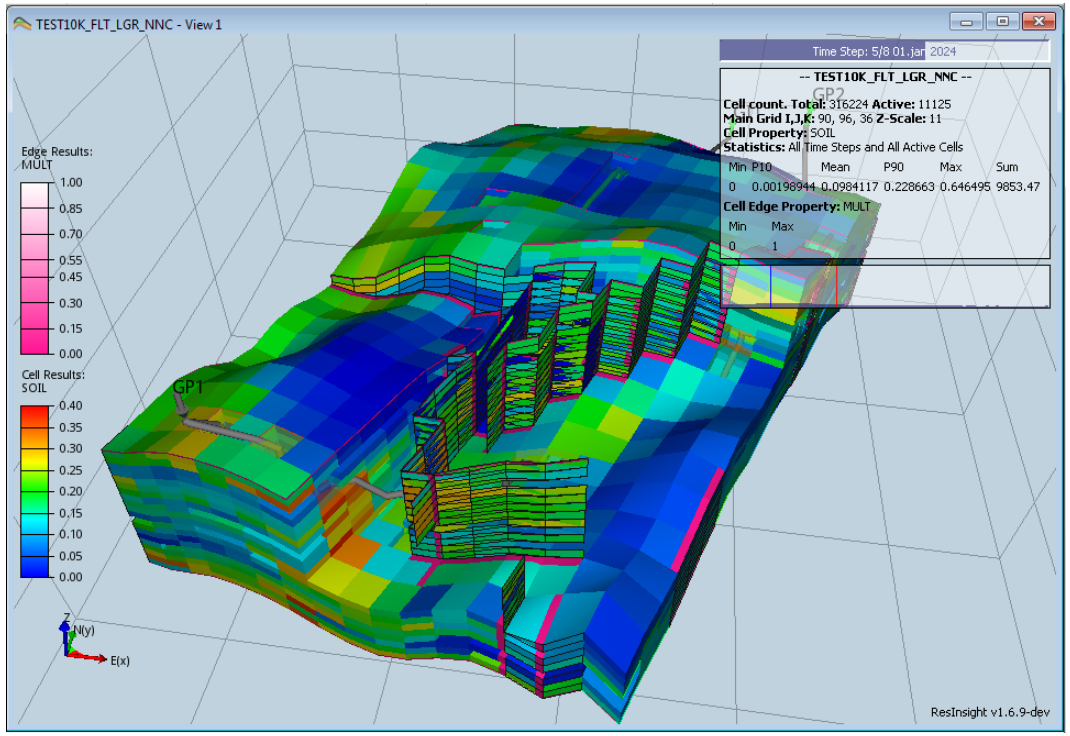
Integrating and covering more of the toolchain

- Integrate with ResInsight?
- More flow diagnostics/reduced models?
- Open geomodeling software?

Collaboration

- Current approach works. Scale to many more contributors?

Offering commercial support?



SINTEF

OPM Flow in MSO4SC

SINTEF

The ideal Flow user (not developer)

- Expert in scientific computing
  - Juggles libraries and compilers
  - Solves mysterious build errors
  - Knows CMake intimately
- Mathematician
  - Knows numerical methods
  - Understands what to do when it does not converge
  - Understands the limitations and errors
- Domain expert in reservoir simulation
  - Knows the why, not just the what
  - Understands the underlying processes
- Elite engineer
  - Knows the input deck format by heart
  - Can coax the simulator to do things it was not designed to do

The ideal Flow user (not developer)

- ~~Expert in scientific computing~~
  - ~~Juggles libraries and compilers~~
  - ~~Solves mysterious build errors~~
  - ~~Knows CMake intimately~~
- Mathematician
  - Knows numerical methods
  - Understands what to do when it does not converge
  - Understands the limitations and errors
- Domain expert in reservoir simulation
  - Knows the why, not just the what
  - Understands the underlying processes
- Elite engineer
  - Knows the input deck format by heart
  - Can coax the simulator to do things it was not designed to do

!

SINTEF

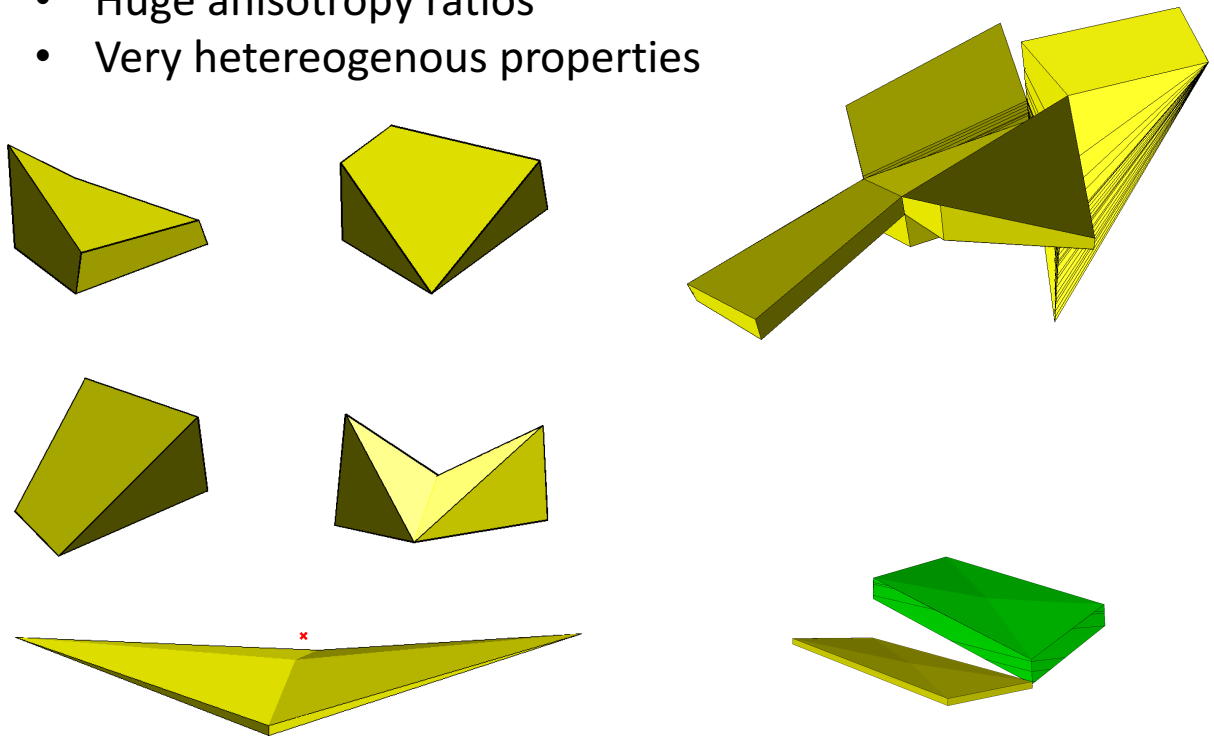
Why MSO4SC

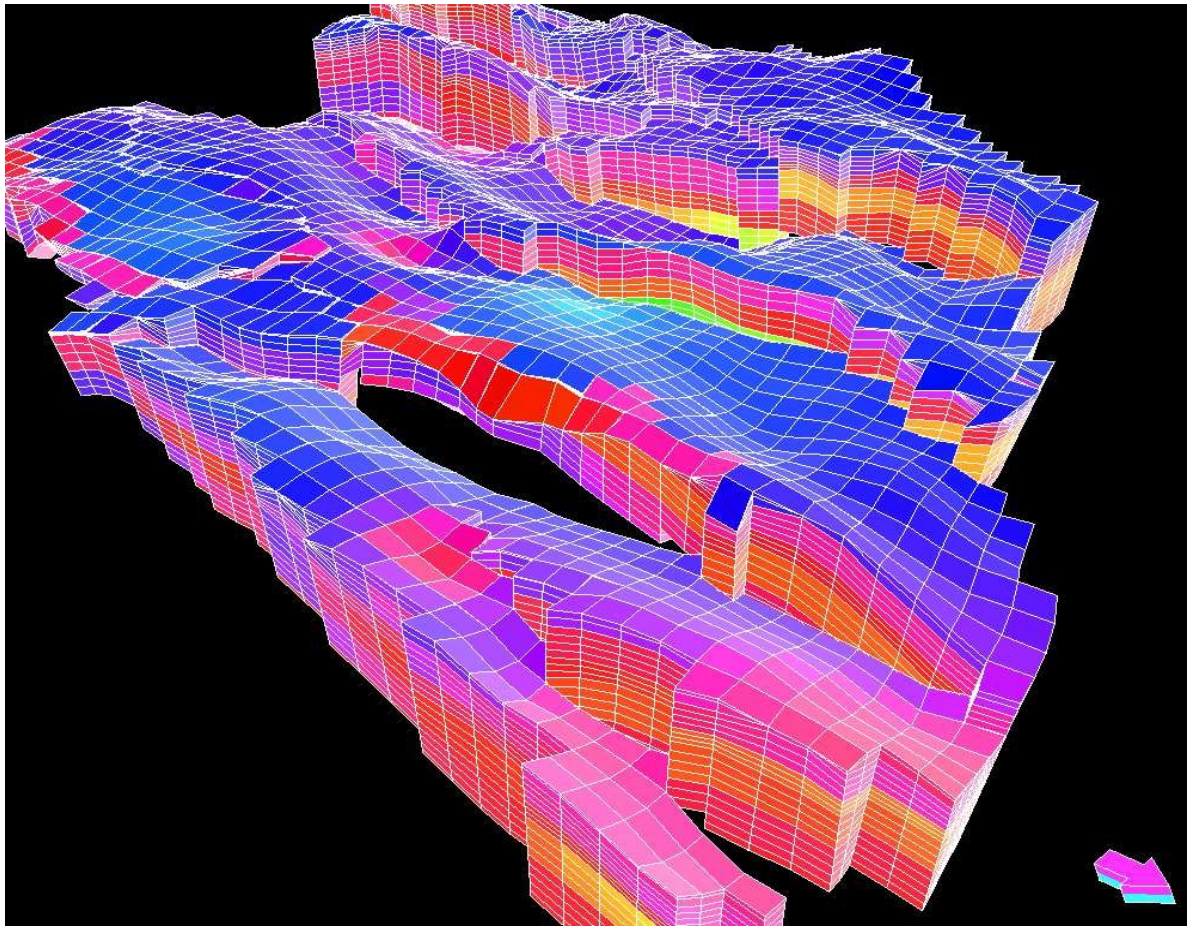
- Take OPM to where usage is going to be
  - Cloud, ensembles, larger scales
- Improved visibility and dissemination
  - Get more users
  - Get more feedback
  - Get more contributors
  - Gain new clients for our services
- Improve OPM software
  - Deployability
  - Usability
  - Scalability

A problem with grid interfaces

Reservoir grids are ”bad”

- Bad cell shapes
- Arbitrary many connections
- Huge anisotropy ratios
- Very hetereogenous properties





How do we write our space discretizations?

Flow: FV order 1, upwind weighting

Requires:

- Connectivity graph
- Transmissibilities on graph edges
- Cell depths and volumes

Ideal grid interface: only the above

High flexibility:

- Manipulate transmissibilities (faults)
- Manipulate connectivity graph (fake aquifers)
- Agnostic to actual grid type (CP, PEBI etc.)

Upscaling: mimetic method

Requires:

- Grid that is a cell-complex
- Interface areas and centroids
- Cell volumes and centroids

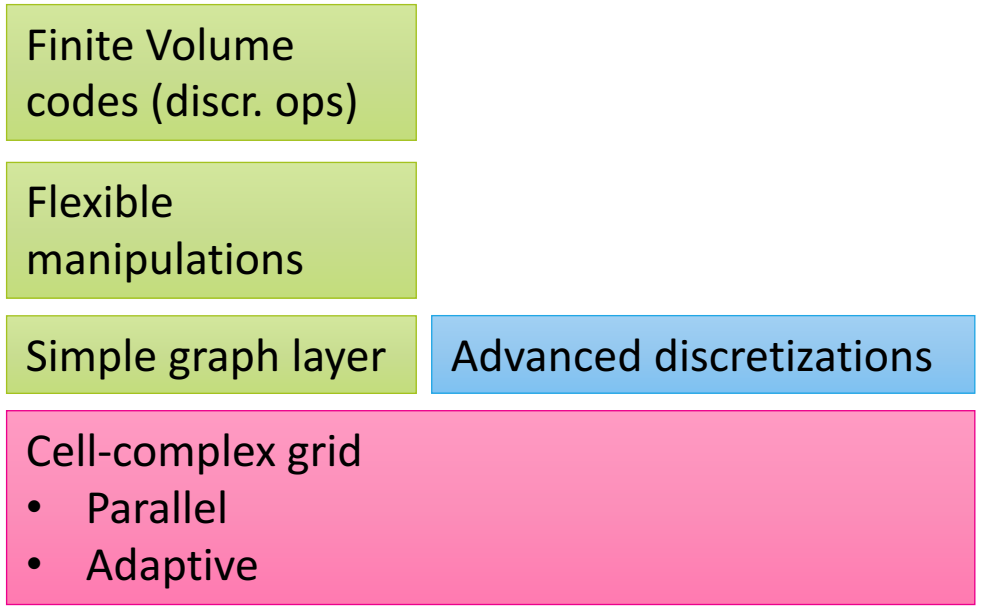
Ideal grid interface: a cell-complex interface

Can support other discretizations:

- Higher-order methods
- Streamline methods
- Virtual Element methods (and some FE)

How can we eat our cake and have it too?

Sketch of an idea:



Example problems with this:

- Implement equation once, yet have discretization flexibility?
- Manipulations restrict adaptivity or vice versa

Collaboration joys and pains

Costs of collaboration

Code quality issues

- No unified coding standard
- Risk of code duplication, maintainance headaches
- Inelegant mixing of approaches and philosophies

Bureaucracy

- *Pull Request* workflow is nontrivial
- Code review is time-consuming
- GitHub discussions can derail to center on unimportant issues

Focus

- Different goals among collaborators
- Research vs. Industry

What we do about it

Find consensus on long-term goals

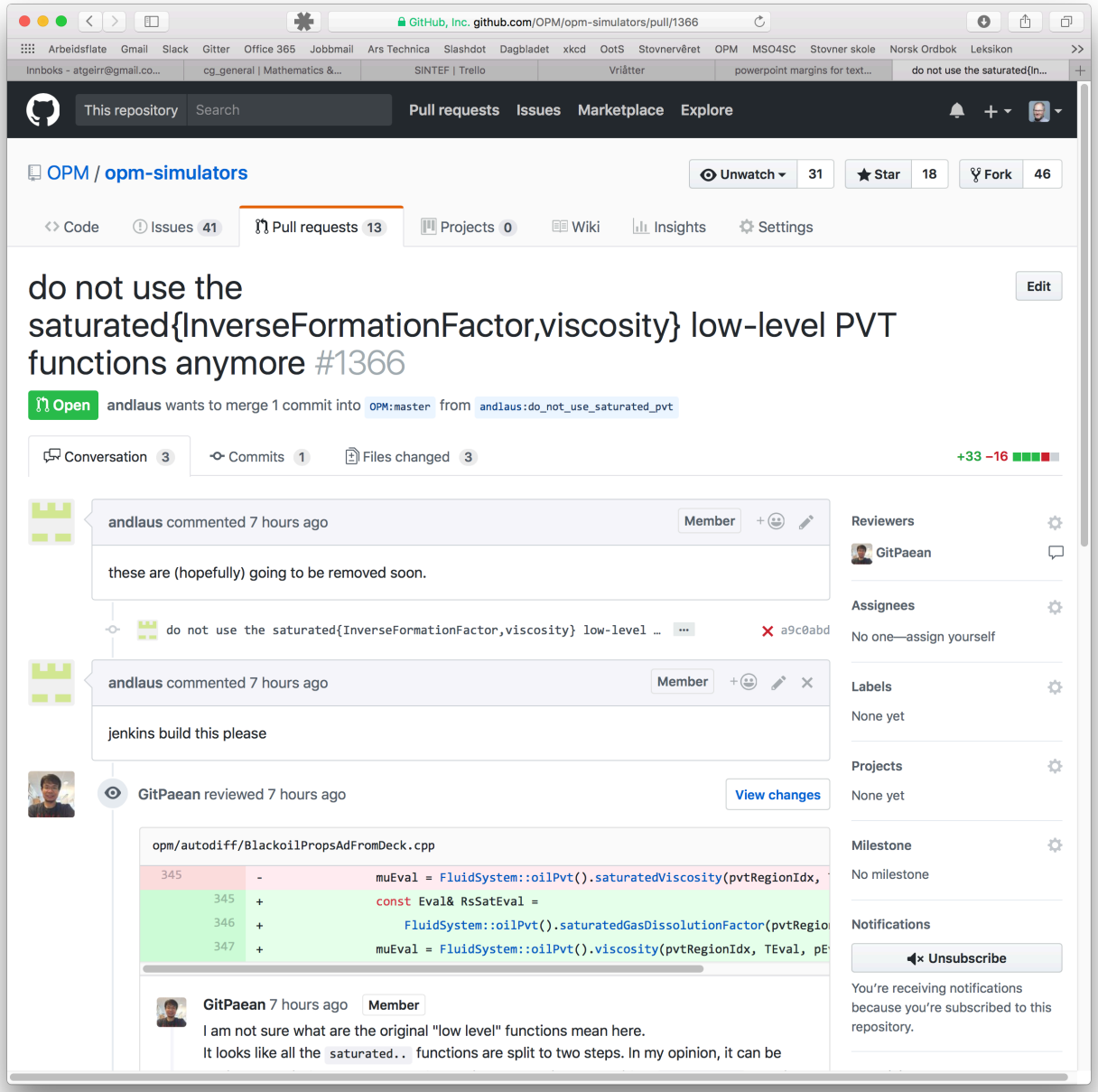
- We want OPM to succeed
- We agree that industrial relevance is key

Communication

- Weekly video meetings
- GitHub Pull Requests are actively discussed
- OPM meetings and other face-to-face meetings

Professional approach to development

- Seek courtesy and good tone (sometimes we fail)
- Automated testing (unit tests, integration tests)



SINTEF

More than the sum of its parts

OPM could never have reached its current state without

- SINTEF (grids, discretizations, numerics, MRST)
- IRIS (robustness, testing, making it converge)
- Statoil (I/O code, focus, funding)
- Individual contributors and Dune project devs
  (local AD assembly, linear solvers, parallel approach)

None of the above could have made it all by themselves!

Making Flow perform well

What is the main bottleneck?

A. Assembly of nonlinear equations?
B. Solving linear systems?
C. Input/output?
D. Other things?

Answer changes over time!

For OPM Flow and our target problems, always A or B.

(I/O performance has also been improved 3x)

Image from Joel McKelvey

Bottleneck 1

Linear solver horrendously slow

- UMFPACK, direct solver
- Works for very small systems (SPE1)
- Breaks down for a few thousand cells

Root cause: linear system not well suited for direct solver

Root cause: direct solvers do not scale well

	Pressure	Water sat	Gas mix/s	Well flux	Well bhp
Conserve W	W_p	W_{sw}	W_x	W_q	W_{bhp}
Conserve O	O_p	O_{sw}	O_x	O_q	O_{bhp}
Conserve G	G_p	G_{sw}	G_x	G_q	G_{bhp}
Well flow	Q_p	Q_{sw}	Q_x	Q_q	Q_{bhp}
Well control				C_q	C_{bhp}

Bottleneck 1 – addressed

Use Schur complement to eliminate well unknowns

Use iterative solvers from Dune

Use 2-stage CPR preconditioner

- Solve almost-elliptic system for pressure (with AMG precond.)
- Solve full system with ILU0 precond.

Results:

- SPE9 runtime 3 minutes (was 30 min)
- Norne case ~6 times Eclipse runtime

	Pressure	Water sat	Gas mix/s	Well flux	Well bhp
Conserve W	W_p	W_{sw}	W_x	W_q	W_{bhp}
Conserve O	O_p	O_{sw}	O_x	O_q	O_{bhp}
Conserve G	G_p	G_{sw}	G_x	G_q	G_{bhp}
Well flow	Q_p	Q_{sw}	Q_x	Q_q	Q_{bhp}
Well control				C_q	C_{bhp}

Figure: Schur complement eliminates well unknowns

Bottleneck 2

Assembly of nonlinear equations slow

- Functions implement residual equations
- AD class produces Jacobians

Root cause: simple operations too expensive

- Every +-*/ op triggers sparse matrix creation
- Even when matrix is diagonal or identity!

```
flux[phase] = upwind.select(b * mob) * (transi * dh);
```

Every multiplication, assignment and select() trigger sparse matrix creation.

Bottleneck 2 – addressed

Replace SparseMatrix in AD class with smart wrapper

- Wrapper treats zero, identity and diagonal matrices with custom code
- No change at all to the simulation code!

Result:

- Norne case ~3.5 times Eclipse runtime

```
flux[phase] = upwind.select(b * mob) * (transi * dh);
```

Now only select() trigger sparse matrix creation (since result depends on unknowns in multiple cells)

Bottleneck 3

Linear solver dominates runtime (again)

- Time-consuming setup of matrices for preconditioner and solver
- Outer linear solve of full system is slow

	Pressure	Water sat	Gas mix/s	Well flux	Well bhp
Conserve W	W_p	W_{sw}	W_x	W_q	W_{bhp}
Conserve O	O_p	O_{sw}	O_x	O_q	O_{bhp}
Conserve G	G_p	G_{sw}	G_x	G_q	G_{bhp}
Well flow	Q_p	Q_{sw}	Q_x	Q_q	Q_{bhp}
Well control				C_q	C_{bhp}

Bottleneck 3 – addressed

Change system matrix structure

- Use block-ILU0 instead of CPR
- Before: 3x3 system of NxN sparse matrices
- Now: NxN sparse matrix of 3x3 blocks
  (or 4x4 for polymer etc.)

Result:

- Norne case ~2.5 times Eclipse runtime

	Pressure	Water sat	Gas mix/s	Well flux	Well bhp
Conserve W	W_p	W_{sw}	W_x	W_q	W_{bhp}
Conserve O	O_p	O_{sw}	O_x	O_q	O_{bhp}
Conserve G	G_p	G_{sw}	G_x	G_q	G_{bhp}
Well flow	Q_p	Q_{sw}	Q_x	Q_q	Q_{bhp}
Well control				C_q	C_{bhp}

Bottleneck 4

Assembly of residual and Jacobians dominate (again)

Root cause: cache-unfriendly use of AD class

Root cause: (still) too many sparse matrix ops

```
flux[phase] = upwind.select(b * mob) * (transi * dh);
```

The multiplication ”`b * mob`” requires writing the result vector to memory before doing the next operation

Bottleneck 4 – addressed

Completely change assembly approach to use local AD

- Meaning: only handle fixed number of local derivatives for each variable
- Much better cache performance
- Matrix assembly is separate
- Clever trick to get derivatives for fluxes (that depend on two cells)
- Was gradually prototyped by A. Lauser for 2-3 years before switching

Results:

- Norne case ~1.7 times Eclipse runtime (~1.1 by now)

Consequences:

- Assembly no longer resembles MRST
- More complex code structure to understand for programmers

Technology for a better society