SINTEF

Accelerating MRST simulations with Julia

Olav Møyner

Technology for a better society

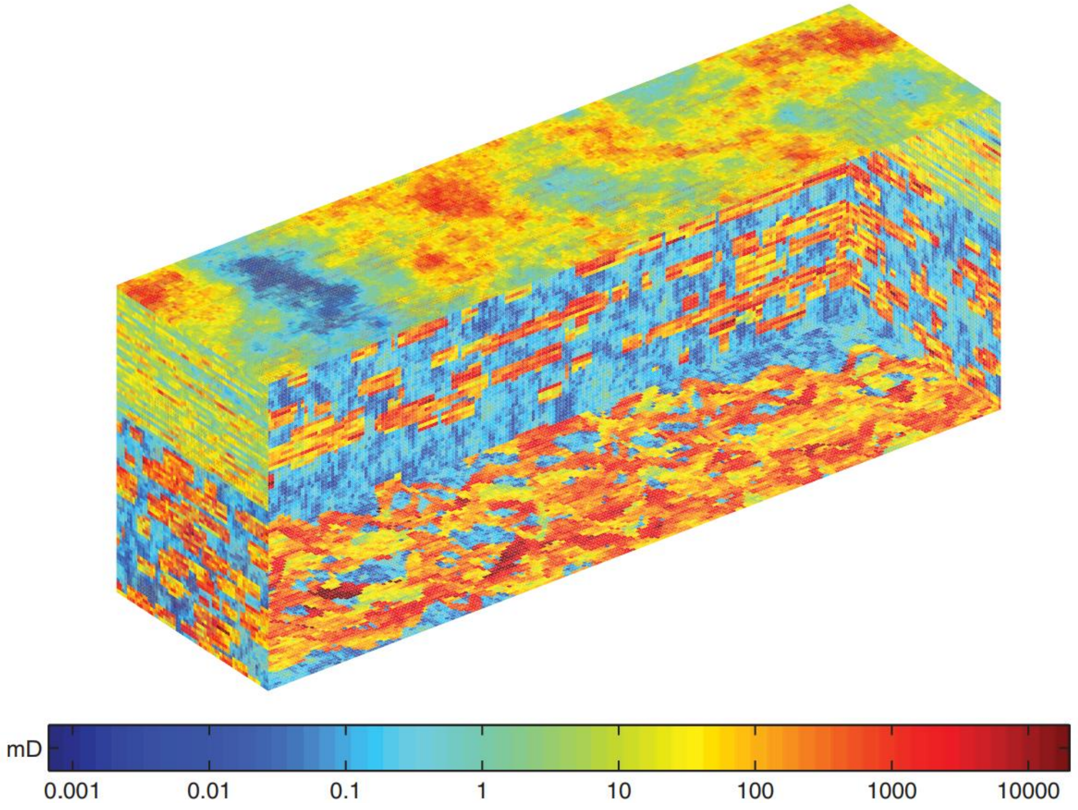
What is Terv & Julia

- Julia is an open-source programming language with syntax similar to Matlab and Python, but with the performance of C++ or Fortran
- https://julialang.org/

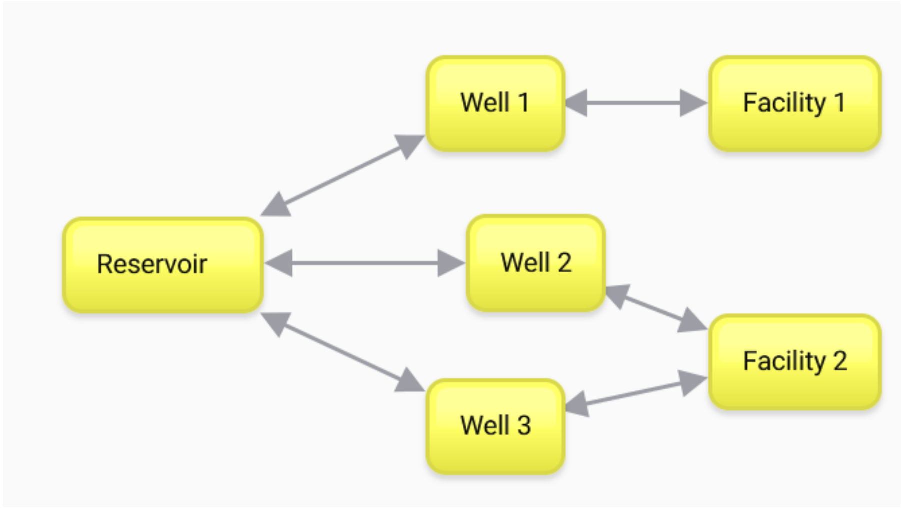
- Terv is a compact, AD-based Julia code suitable for reservoir simulators heavily inspired by MRST
  - Main code 5000 loc
  - Reservoir simulator with MS wells: 1800 lines
  - Support for grids and wells made in MRST
- Goal: Assess potential of Julia for reservoir simulation and other MRST applications
- MRST accelerator: Build cases in MRST and execute on CPU or GPU through Julia!
- Can run immiscible MRST cases directly

Example: SPE10, model 2

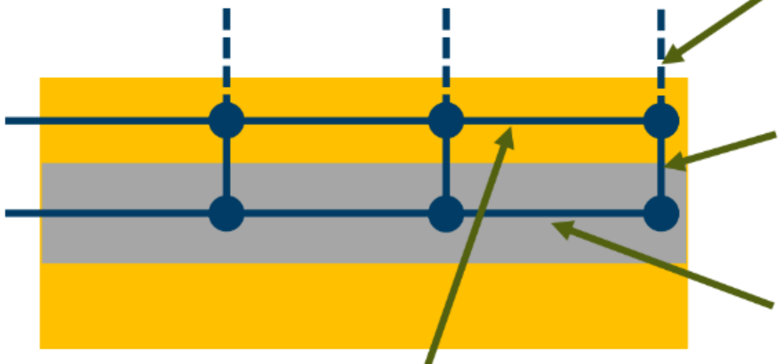
- Benchmark model
  - *Tenth SPE comparative solution project: A comparison of upscaling techniques, Blunt & Christie, SPE REE, 2001*
- 1 122 000 cells
- Single-phase flow, sources
- MRST assembly: 230 ms
- Julia assembly: 75 ms (**3x**)
- Julia assembly (GPU, GTX 1080):
  - Float64: 13 ms (**17.7x**)
  - Float32: 10 ms (**23.0x**)
- Vectorized Matlab code is efficient here



Modelling of wells

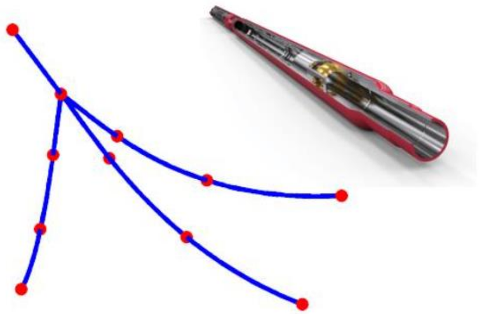


- Full support for multisegment wells
- Based on MRST's multisegment wells:
  - Rigorous conservation law in each well
  - Support for different controls
  - Degrees of freedom:
    - Same as reservoir in nodes, total mass rate on each segment
  - Facility: Total mass rate to surface conditions.



$$\frac{\partial}{\partial t}(\rho_\alpha S_\alpha) + \nabla \cdot (x_\alpha \vec{V}_{mix}) = q, \ \alpha \in \{l, v\}$$

$$\nabla p - g\rho_m \nabla z - h(\vec{V}_m, \mu_m, \rho_m) = 0, \quad g_m = \sum_\alpha S_\alpha g_\alpha$$

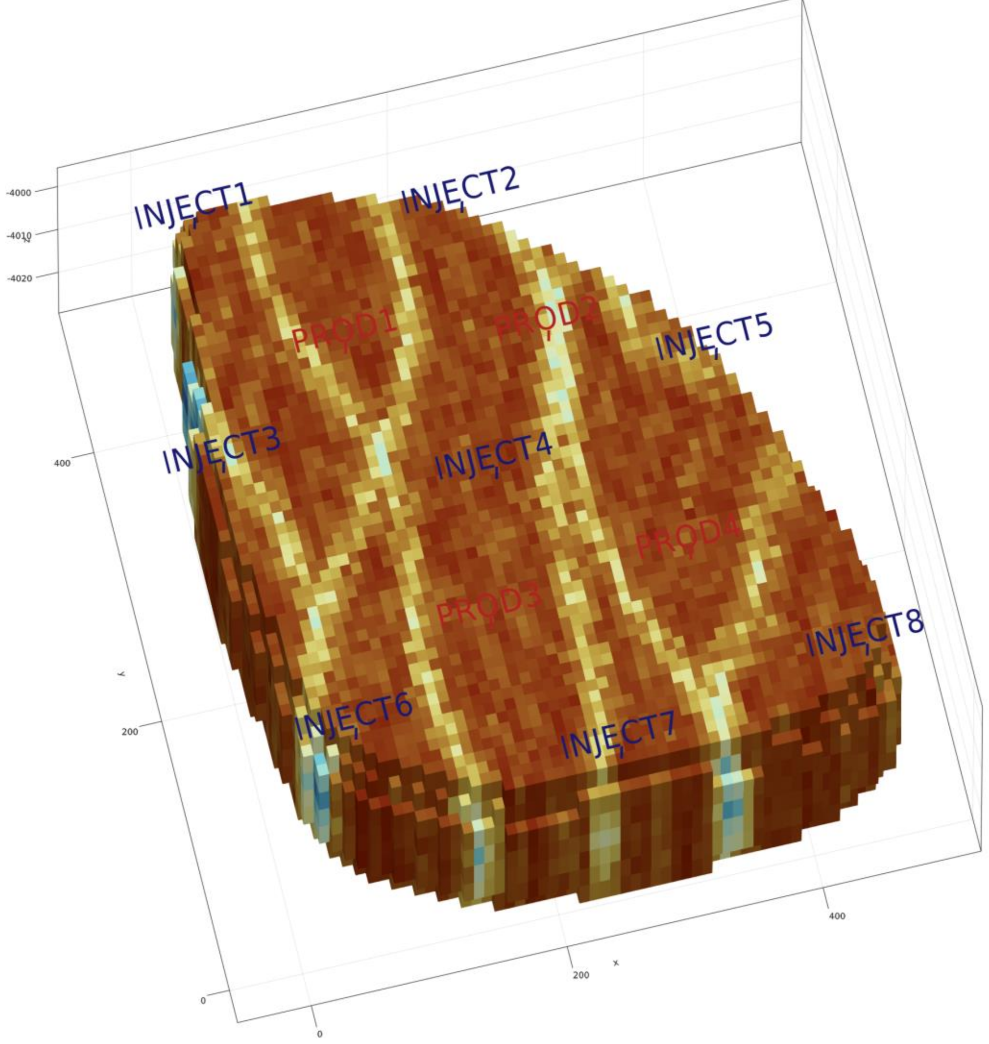


Example: Egg model

- Black-oil model in Eclipse format
- Compressible two-phase water-oil (dead oil)
- 18553 cells, 2 variables each
- Flow driven by multisegment wells
  - 8 wells are injecting water at fixed rate
  - 4 producers operating at fixed pressure
- 3600 days of operation over 123 time-steps
- Julia implementation of two stage CPR preconditioner for Krylov solver
  - Block ILU(0) + AlgebraicMultigrid.jl

The egg model – a geological ensemble for reservoir simulation

J.D. Jansen et al, Geoscience Data Journal, 2014



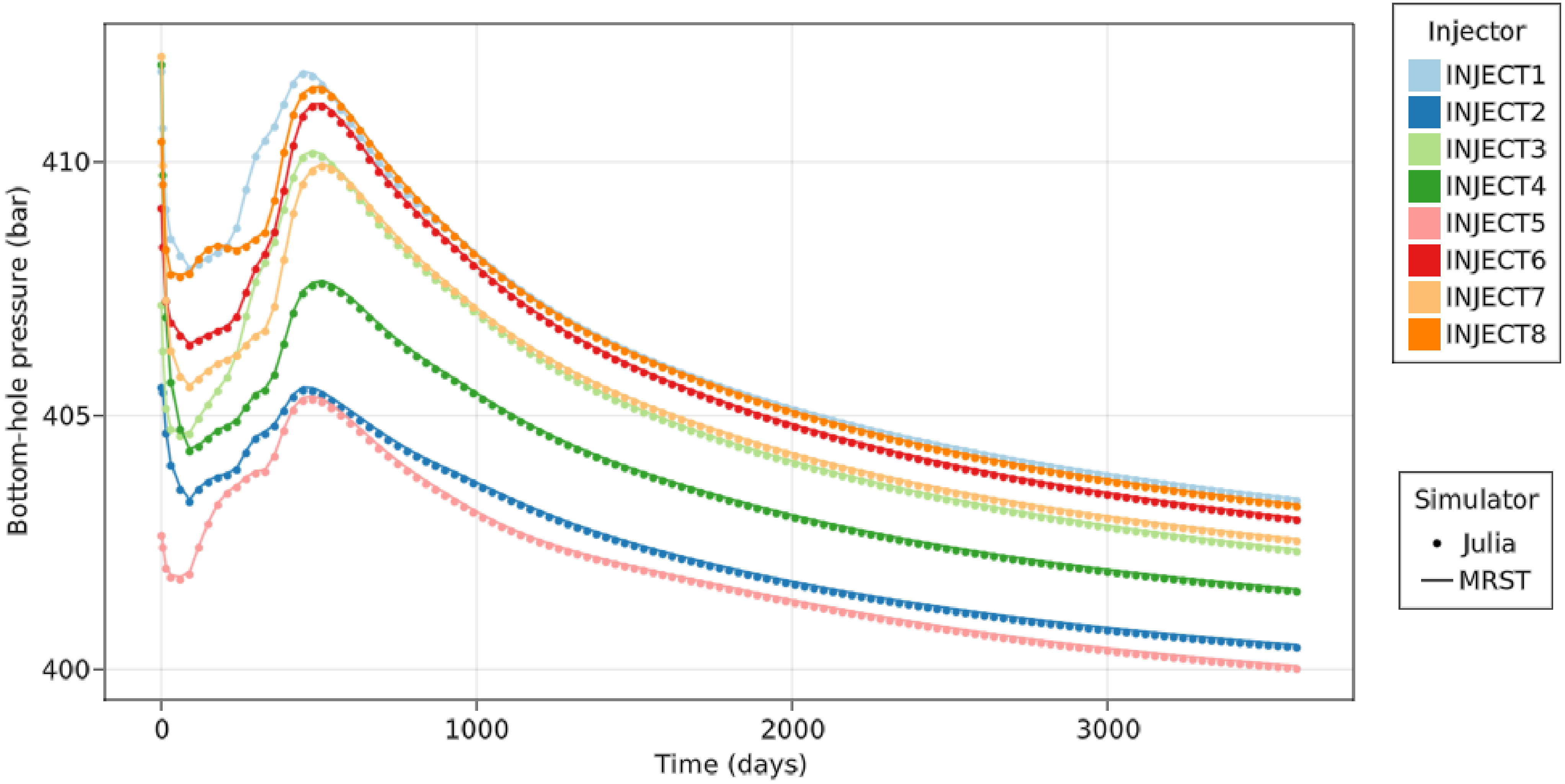

```
# IncompleteLU preconditioner ILUZero.jl or CuSPARSE
ilu = LUPreconditioner()
# AMG preconditioner (AlgebraicMultigrid.jl)
amg = AMGPreconditioner(smoothed_aggregation)
# CPR preconditioner with AMG for first stage and ILU(0) for second
cpr = CPRPreconditioner(amg, ilu, strategy = :true_impes)

# Use Krylov.jl for dqgmres
lsolve = GenericKrylov(dqgmres, verbose = 0, preconditioner = prec,
                        relative_tolerance = 1e-3)
```

Example: Egg model

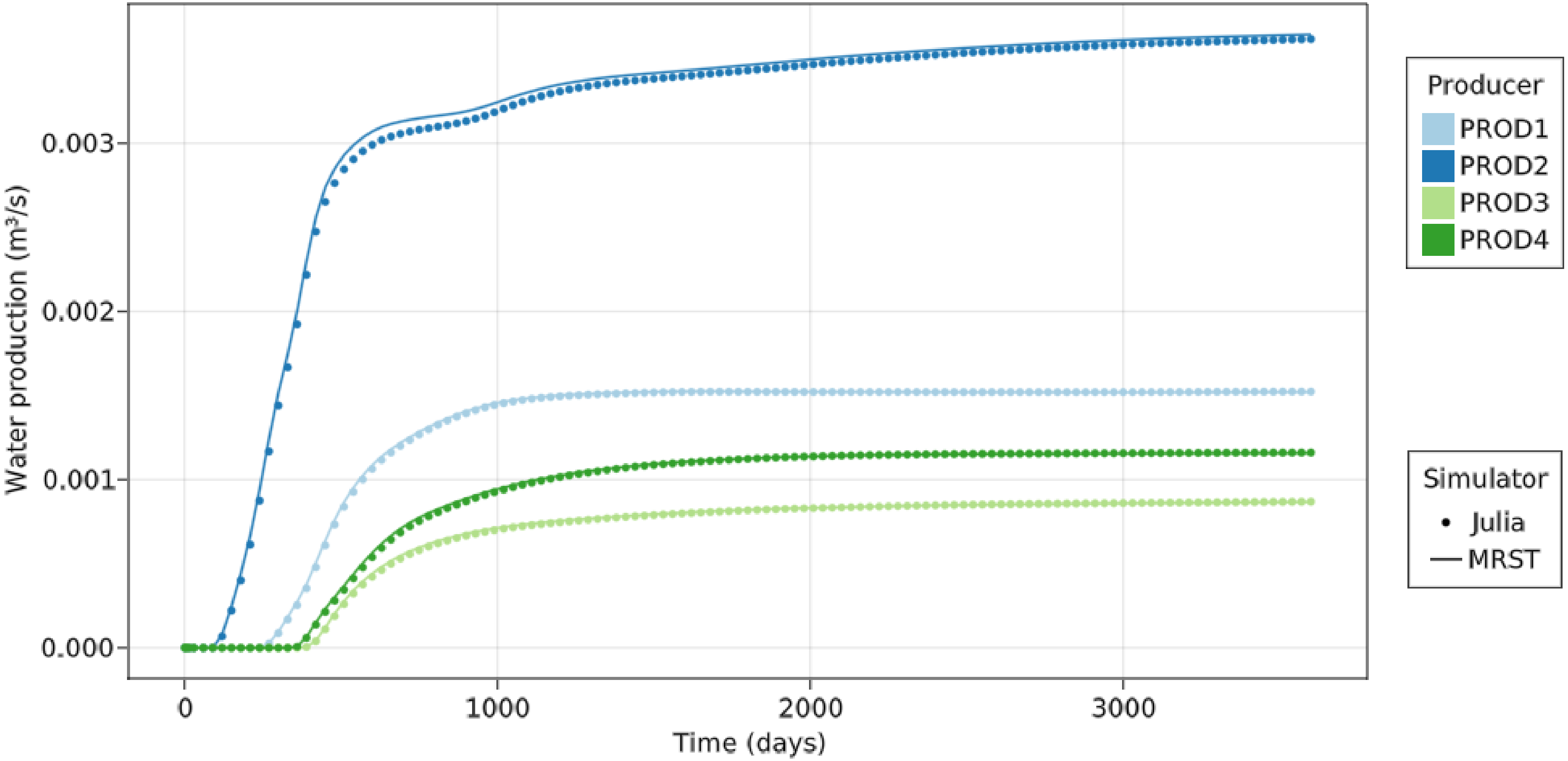
Model	Equation	‖R‖	ϵ
Reservoir	mass_conservation	7.8994e-06 7.7426e-06	1.0000e-02
INJECT1	potential_balance	3.9227e-02	1.0000e-03
	mass_conservation	1.0020e-02 7.6085e-03	1.0000e-02
Facility	control_equation	6.2463e+05	1.0000e-03

```
[ Info: Starting simulation
[ Info: Solving step 1/123 of length 1 day.
[ Info: Solving step 2/123 of length 4 days.
[ Info: Solving step 3/123 of length 1 week, 3 days.
[ Info: Solving step 4/123 of length 2 weeks, 1 day.
[ Info: Solving step 5/123 of length 4 weeks, 2 days.
[ Info: Solving step 6/123 of length 4 weeks, 2 days.
[ Info: Solving step 7/123 of length 4 weeks, 2 days.
[ Info: Solving step 8/123 of length 4 weeks, 2 days.
[ Info: Solving step 9/123 of length 4 weeks, 2 days.
[ Info: Solving step 10/123 of length 4 weeks, 2 days.
[ Info: Solving step 11/123 of length 4 weeks, 2 days.
[ Info: Solving step 12/123 of length 4 weeks, 2 days.
[ Info: Solving step 13/123 of length 4 weeks, 2 days.
[ Info: Solving step 14/123 of length 4 weeks, 2 days.
[ Info: Solving step 15/123 of length 4 weeks, 2 days.
```

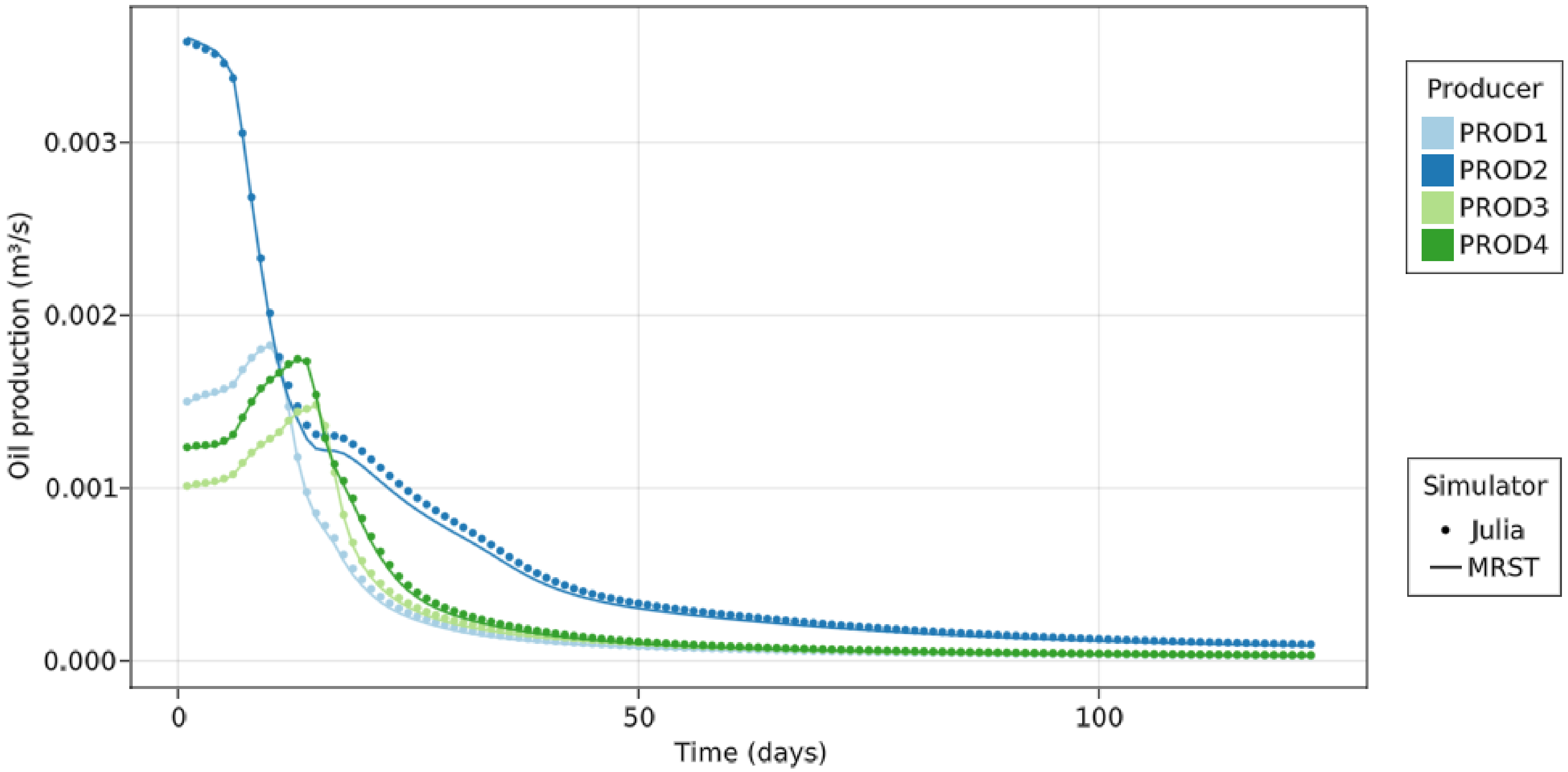
Example: Egg model



Example: Egg model



Example: Egg model



Example: Egg model

- Compared against MRST (equilibrium wells):
  - Both codes use two stage CPR preconditioner with block ILU(0) and AMG
  - Speedup per assembly (Matlab with C++ acceleration): **10x**
  - Speedup per assembly (Matlab only): **14x**
  - AMGCL C++ CPR code gives similar performance
  - **Total speedup of 4** (linear solve takes up 60%)
- Compared against OPM Flow (MS wells):
  - Speedup assembly: **2x**
  - **Total speedup 1.5x**
- **Caveats:**
  - Single-threaded results
  - Many settings to tune – OPM uses default settings
  - OPM used different linear solver

Number of iterations

Type	Per step #123	Per ministep #123	Total
newtons	4.47967	4.47967	551
linearizations	5.47967	5.47967	674

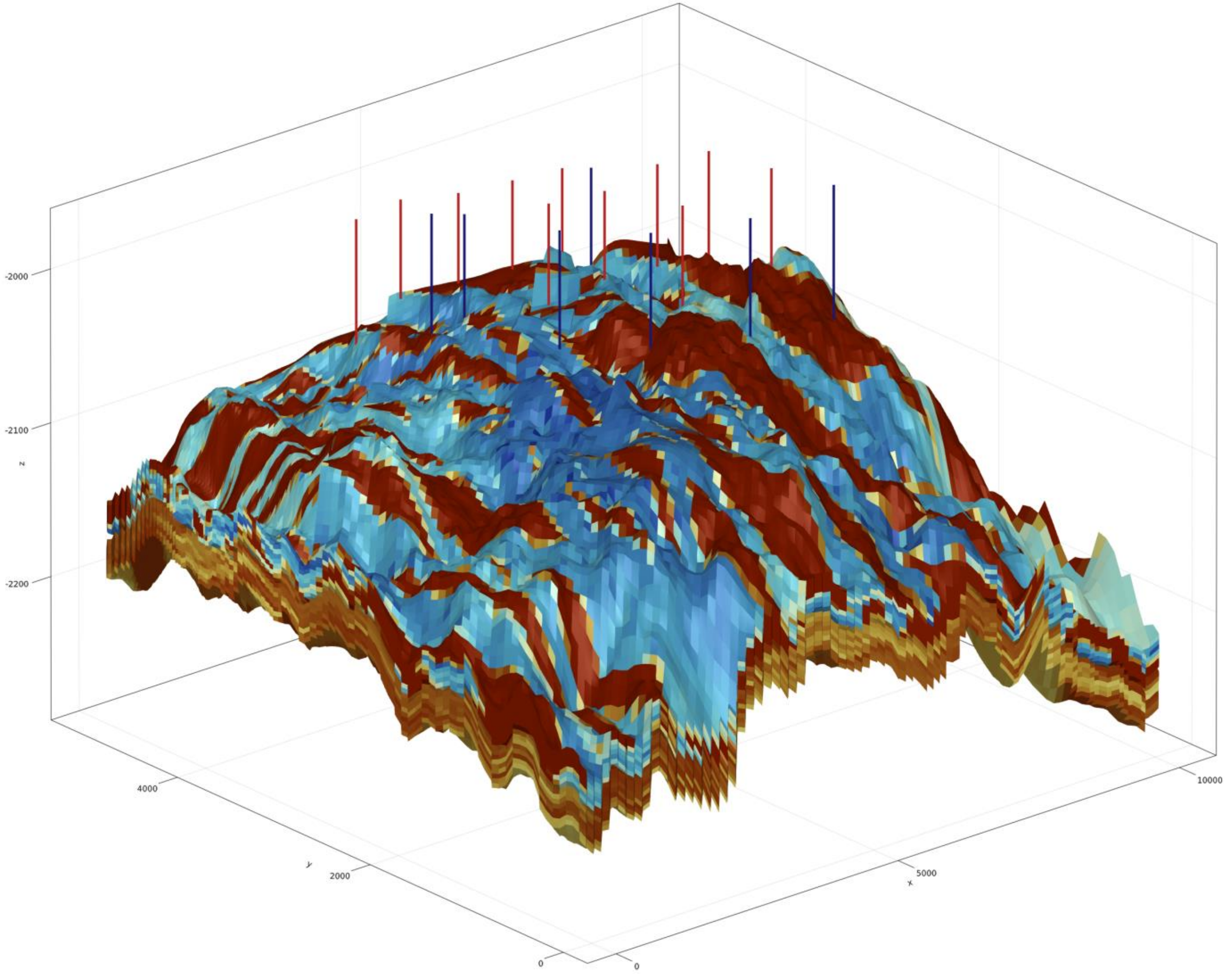
Simulator timing

Type	Each seconds	Total %	Total seconds
assembly	6.55e-03	19.28	4.41
linear_system	2.88e-03	8.48	1.94
linear_solve	2.64e-02	63.58	14.55
update_time	2.53e-03	6.09	1.39
convergence	5.94e-04	1.75	0.40
other	3.40e-04	0.82	0.19
total	4.15e-02	100.00	22.88

Olympus model

- 192 749 cells
- Dead-oil model (similar to Egg)
- 18 wells
- Single thread:
  - Julia: 96 ms per assembly
  - Matlab: 720 ms (**7.5x**)
  - OPM Flow: 216 ms (**2.25x**)

Overview of the Olympus field development optimization challenge, RM Fonseca et al, ECMOR XVI 2018



SINTEF

Technology for a better society