Reordering nonlinear solver in OPM Flow

Atgeirr Flø Rasmussen
Øystein Klemetsdal

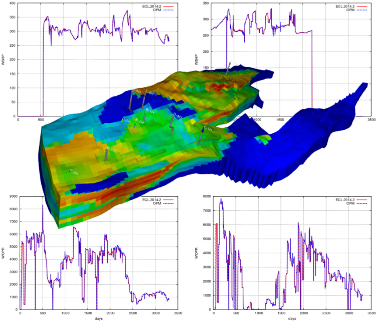
SINTEF Digital, Mathematics and Cybernetics

OPM Meeting 2017
Bergen, October 17, 2017

Overview of talk

OPM Flow at a glance

- Open source
- Competitive performance
- Full industrial complexity
    - Well controls
    - EOR: CO2, polymer
    - CO2 sequestration
- Automatic differentiation (AD)



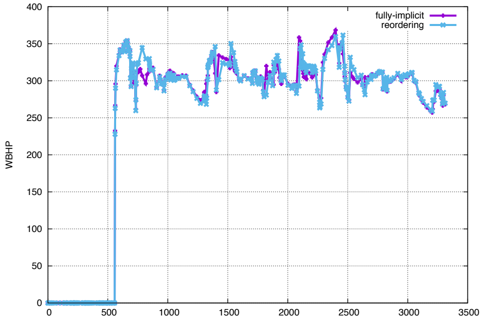
Ambition: to be a strong base for both industrial development and academic research

Main idea of this talk

We can gain performance by

- sequential splitting,
- reordering solvers,
- (nonlinear preconditioners)

... without losing the ability to run *industrial full field models*.



Case: Norne
Well: C-3H
Plot: BHP

The black-oil model, physical laws

"Black-oil" model assumptions

Lump hydrocarbon species into two pseudo-components (oil, gas)

Fluid	Pseudo-component		
Phase	Water	Oil	Gas
Aqueous	x		
Oleic		x	x
Gaseous		x	x

(Subset of more general *compositional* model)

Conservation of mass (per component α)

$$\frac{\partial}{\partial t}(\phi A_\alpha) + \nabla \cdot \mathbf{u}_\alpha = Q_\alpha$$

Darcy's law (per phase β)

$$\mathbf{v}_\beta = -(k_{r,\beta}/\mu_\beta)\mathbf{K}(\nabla p_\beta - \rho_\beta \mathbf{g})$$

System of equations (fully implicit)

System of PDEs, one for each pseudo-component α:

$$\frac{\partial}{\partial t}\left(\phi A_\alpha\right) + \nabla \cdot \mathbf{u}_\alpha = Q_\alpha$$

where

$$\begin{aligned} A_w &= b_w s_w, & \mathbf{u}_w &= b_w \mathbf{v}_w, \\ A_o &= b_o s_o + r_V b_g s_g, & \mathbf{u}_o &= b_o \mathbf{v}_o + r_V b_g \mathbf{v}_g, \\ A_g &= b_g s_g + r_S b_o s_o, & \mathbf{u}_g &= b_g \mathbf{v}_g + r_S b_o \mathbf{v}_o, \end{aligned}$$

and also:

$$\begin{aligned} s_w + s_o + s_g &= 1 \\ p_o - p_w &= p_{cow} \\ p_o - p_g &= p_{cog}. \end{aligned}$$

and Darcy:

$$\mathbf{v}_\beta = -(k_{r,\beta}/\mu_\beta)\mathbf{K}(\nabla p_\beta - \rho_\beta \mathbf{g})$$

Coupling to well model

What is the source term, anyway?

$$\frac{\partial}{\partial t}\left(\phi A_\alpha\right) + \nabla \cdot \mathbf{u}_\alpha = Q_\alpha$$

Coupling to well model

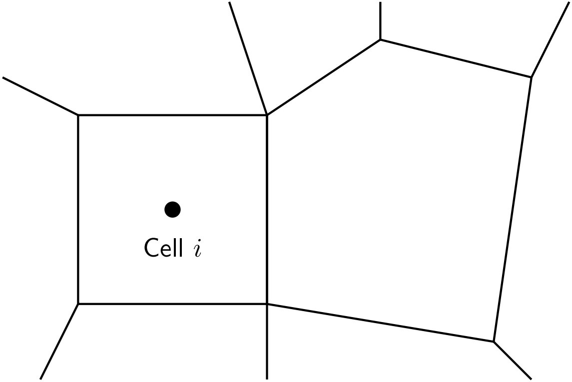
What is the source term, anyway?

$$\frac{\partial}{\partial t}\left(\phi A_\alpha\right) + \nabla \cdot \mathbf{u}_\alpha = Q_\alpha$$

Well rates!

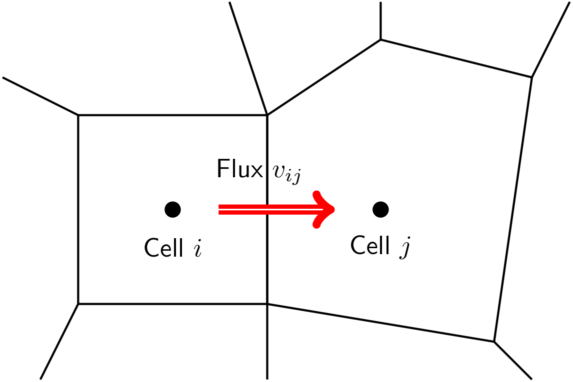
- Computed using separate well model(s).
- Must be solved *simultaneously* with reservoir equations.

Fully implicit discretization (I)



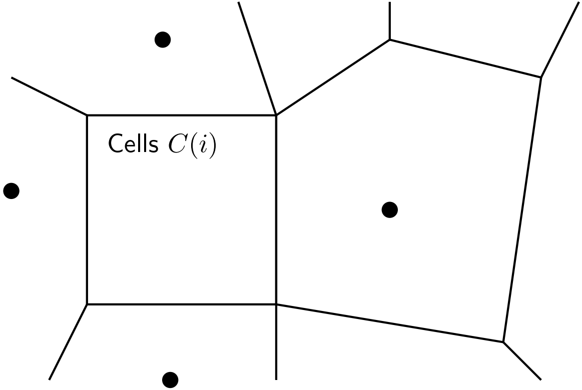
Notation:
Cell value: x_i or x_j
Connection value: x_{ij}

Fully implicit discretization (I)



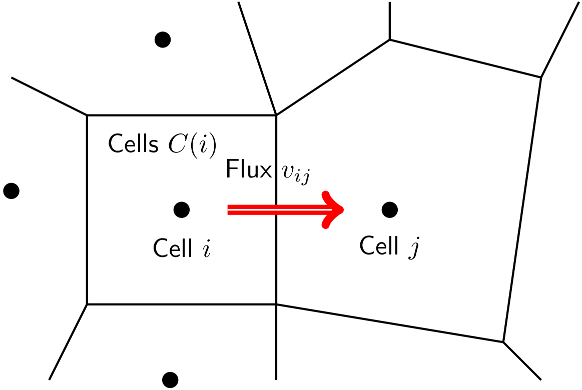
Notation:
Cell value: x_i or x_j
Connection value: x_{ij}

Fully implicit discretization (I)



Notation:
Cell value: x_i or x_j
Connection value: x_{ij}

Fully implicit discretization (I)



Notation:
Cell value: x_i or x_j
Connection value: x_{ij}

Discrete material balance

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

Fully implicit discretization (II)

The discretized equations are, for each pseudo-component α and cell i:

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

where

$$\begin{aligned} A_w &= b_w s_w, & u_w &= b_w v_w, \\ A_o &= b_o s_o + r_V b_g s_g, & u_o &= b_o v_o + r_V b_g v_g, \\ A_g &= b_g s_g + r_S b_o s_o, & u_g &= b_g v_g + r_S b_o v_o. \end{aligned}$$

Also:

$$s_w + s_o + s_g = 1 \qquad p_{cow} = p_o - p_w \qquad p_{cog} = p_o - p_g.$$

Fully implicit discretization (III)

The phase fluxes are computed for each connection ij as follows:

$$
\begin{aligned}
\rho_{\alpha,ij} &= (\rho_{\alpha,i} + \rho_{\alpha,j})/2 \\
\Delta H_{\alpha,ij} &= p_{\alpha,i} - p_{\alpha,j} - g\rho_{\alpha,ij}(z_i - z_j) \\
UP(\alpha, ij) &= \begin{cases} i & \Delta H_{\alpha,ij} \geq 0 \\ j & \Delta H_{\alpha,ij} < 0 \end{cases} \\
(b_\alpha v_\alpha)_{ij} &= (b_\alpha \lambda_\alpha)_{UP(\alpha,ij)} T_{ij} \Delta H_{\alpha,ij} \\
(r_S b_o v_o)_{ij} &= (r_S b_o \lambda_o)_{UP(o,ij)} T_{ij} \Delta H_{o,ij} \\
(r_V b_g v_g)_{ij} &= (r_V b_g \lambda_g)_{UP(g,ij)} T_{ij} \Delta H_{g,ij}
\end{aligned}
$$

(Darcy discretized with TPFA using phase-based upwinding)

Solving the fully implicit discrete equations

Main method: Newton-Raphson

- solve large heterogenous linear systems
- challenging to precondition (CPR + AMG best?)
- must modify updates for phase changes
- must handle convergence failures (timestep cuts)

Sequential splitting

Splitting *pressure* and *transport*:

- lets us solve two smaller problems rather than one large
- allows specialized methods to be used
  - Pressure: multiscale methods
  - Transport: **reordering methods**, streamline methods
- gives rise to splitting error

Important questions

- Can splitting methods be applied to real field cases?
  - Yes!
- Will they yield improved performance for such cases?
  - Yes, probably.
- Acceptable robustness compared to fully implicit methods?
  - Yes, probably.

Recall...

Discrete material balance

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

Sequential implicit discretization

Pressure equation: linear combination to eliminate saturation dep.

$$R_p = \sum_\alpha \sigma_\alpha R_\alpha = 0,$$

$$\sigma_w = 1/b_w, \qquad \sigma_o = \frac{1/b_o - r_S/b_g}{1 - r_S r_V}, \qquad \sigma_g = \frac{1/b_g - r_V/b_o}{1 - r_S r_V}.$$

Store $v_{T,ij} = \sum_\alpha v_{\alpha,ij}$ for transport solver.

Transport equations

$$R_o = 0, \qquad R_g = 0,$$

with fluxes derived from v_T:

$$(b_\alpha v_\alpha)_{ij} = b_{\alpha,ij} \frac{\lambda_{\alpha,ij}}{\sum_\beta \lambda_{\beta,ij}} (v_T + T_{ij} G_{ij}).$$

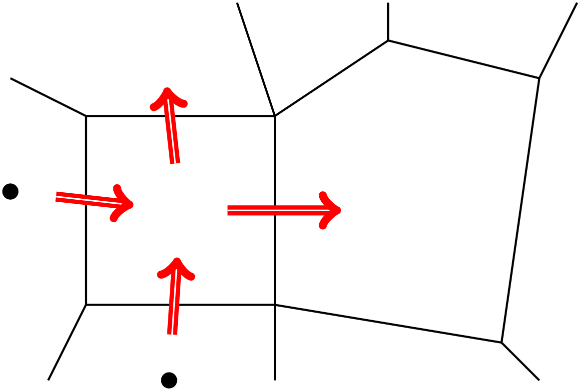
Upwind b_{ij}, λ_{ij} and gravity term G_{ij} following Brenier and Jaffré "Upstream Differencing for Multiphase Flow in Reservoir Simulation"

The reordering idea

Advection-dominated transport problems:

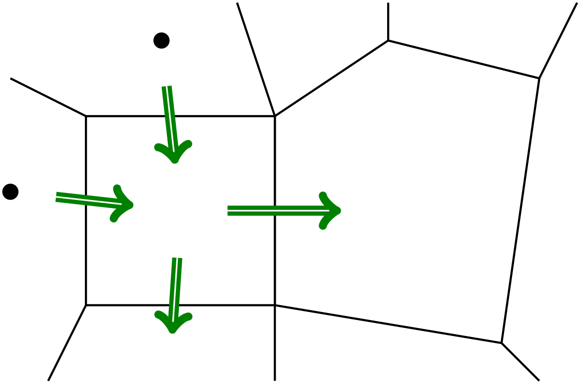
- Upwind discretization
- Information flows in direction of fluid flow
- Solution in upstream cells does not depend on solution in downstream cells
  (... unless we have loops or countercurrent flow)

Discretization setting (transport)



Fluxes $\color{red}{v_{g,ij}}$, $j \in C(i)$.
Cells $U(i, g)$.

Discretization setting (transport)



Fluxes $v_{o,ij}$, $j \in C(i)$.
Cells $U(i,o)$.

Reordering the transport equations

Rewrite transport equation (for cell i, phase α):

$$F_i(x_i) + \sum_{j \in C(i)} G_i(x_i, x_j) v_{ij}(x_i, x_j, v_{T,ij}) = 0$$

v_{ij}: signed flux from cell i to cell j
x_i: unknowns in cell i

With upstream weighting we can write:

$$F_i(x_i) + \sum_{j \in U(i)} G_i^U(x_j) v_{ij}(x_j, v_{T,ij}) + \sum_{j \in D(i)} G_i^D(x_i) v_{ij}(x_i, v_{T,ij}) = 0$$

Given $x_j, j \in U(i)$, we can solve for x_i separately!

Countercurrent flow $\implies$ can only do this per phase
(but we will relax this later)

For 1D case: can solve sequentially from injector to producer!

Newton-Raphson vs. Nonlinear Gauss-Seidel

Newton-Raphson

$r = F(x)$
while $||r|| > tolerance$ **do**
- Compute Jacobian matrix $J = dF/dx$
- Solve $Je = r$
- Update $x = x - e$
- $r = F(x)$

Gauss-Seidel

$r = F(x)$ **while** *any* $||r_i|| > tolerance$ **do**
- **for** *all cells* i **do**
  - Solve single-cell problem i
  - Update x_i
- $r = F(x)$

Perfect ordering: can drop outer loop!

General case: computing an ordering

What quantity to use?

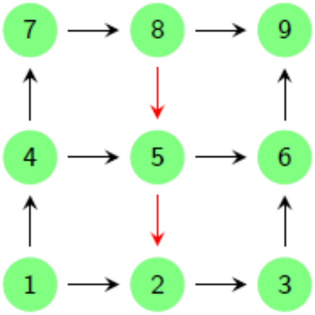
Single-phase flow with no gravity: sort according to pressure.

General case:

- Phase pressure? Which phase?
- Phase fluxes? Which phase?
- Total flux? What about countercurrent flow?

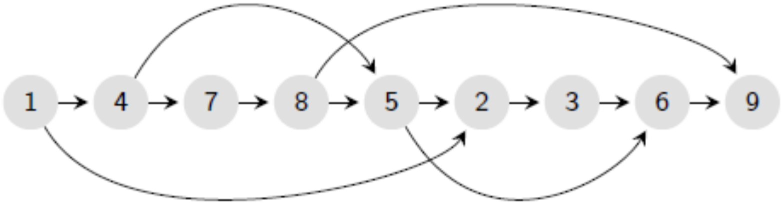
Our answer is: use total flux

Small example



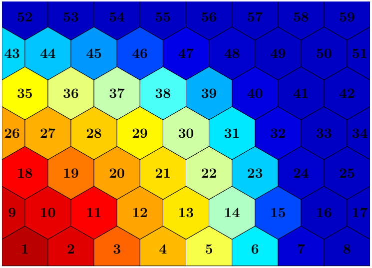
$$V = \begin{bmatrix} \ast & & & & & & & & \\ \ast & \ast & & & {\color{red}\ast} & & & & \\ & \ast & \ast & & & & & & \\ \ast & & & \ast & & & & & \\ & & & \ast & \ast & & & {\color{red}\ast} & \\ & & \ast & & \ast & \ast & & & \\ & & & \ast & & & \ast & & \\ & & & & & & \ast & \ast & \\ & & & & & \ast & & \ast & \ast \end{bmatrix}$$

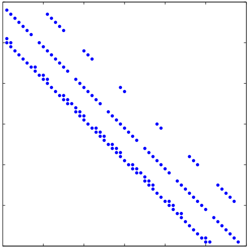
After topological sorting (Tarjan's algorithm): unidirectional graph



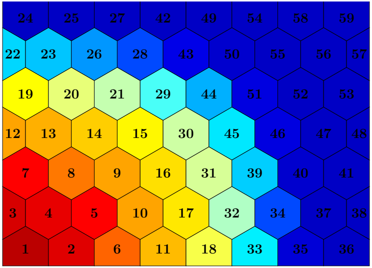
Slightly bigger example

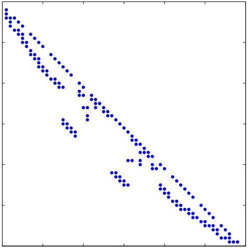
Natural ordering:





After reordering (Tarjan):

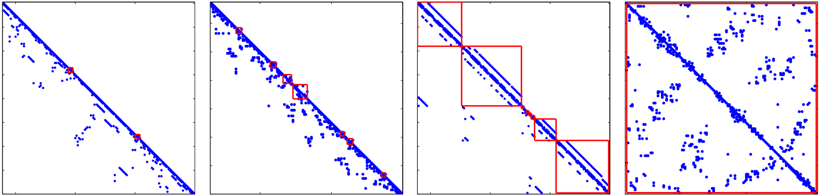




Ordering challenges

- Circular flow (gravity)
- Countercurrent flow (gravity, capillary pressure)

With stronger coupling, more mutually dependent cells:



How to handle such sets (strongly connected components)?

Solution: Gauss-Seidel iterations

Gauss-Seidel

$r = F(x)$ **while** *any* $||r_i|| > tolerance$ **do**

  for *all cells* i **do**

    Solve single-cell problem i

    Update x_i

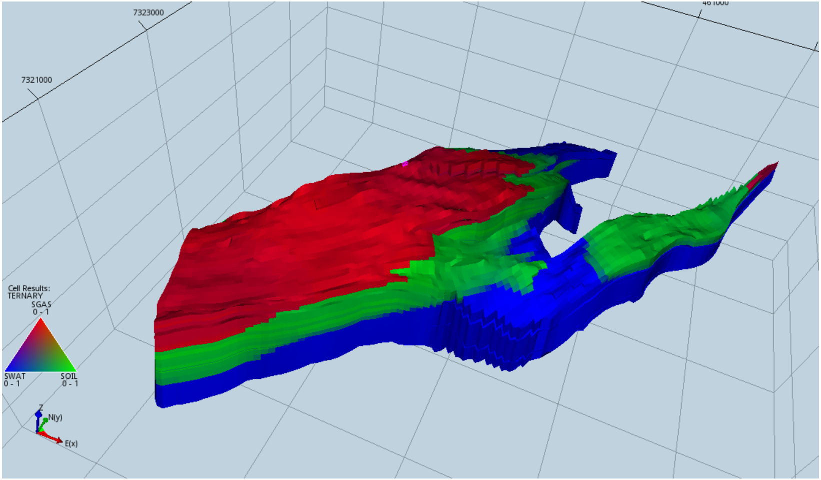
  $r = F(x)$

Apply outer loop, but *only for strongly connected cells*.

Convergence proofs:

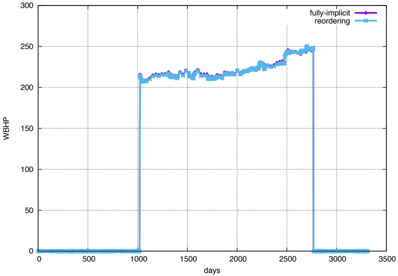
2-phase + polymer	yes
3-phase black-oil	no (but promising numerical results)

Norne field case

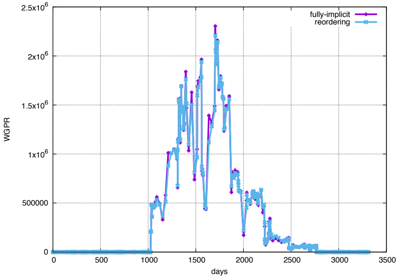


Norne real field case, initial saturation values

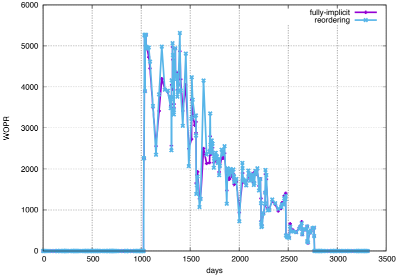
Norne field case: well D-3AH



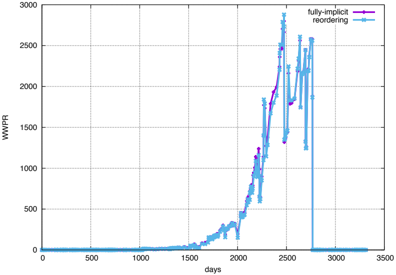
Bottom-hole pressure



Gas production rate

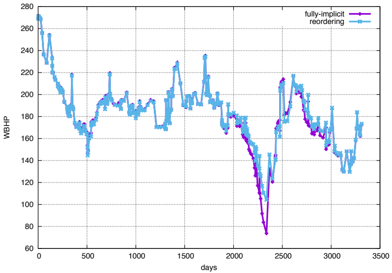


Oil production rate

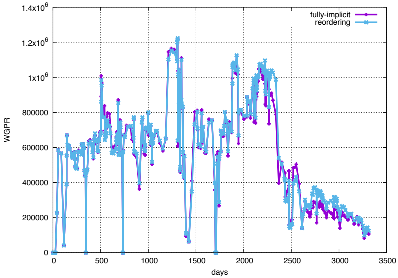


Water production rate

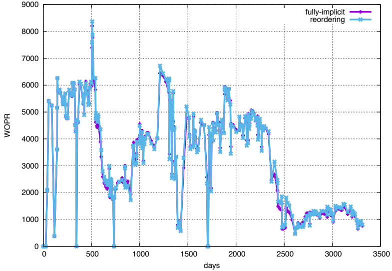
Norne field case: well B-2H



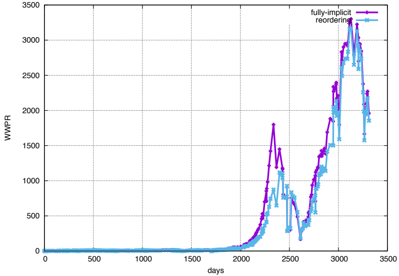
Bottom-hole pressure



Gas production rate



Oil production rate



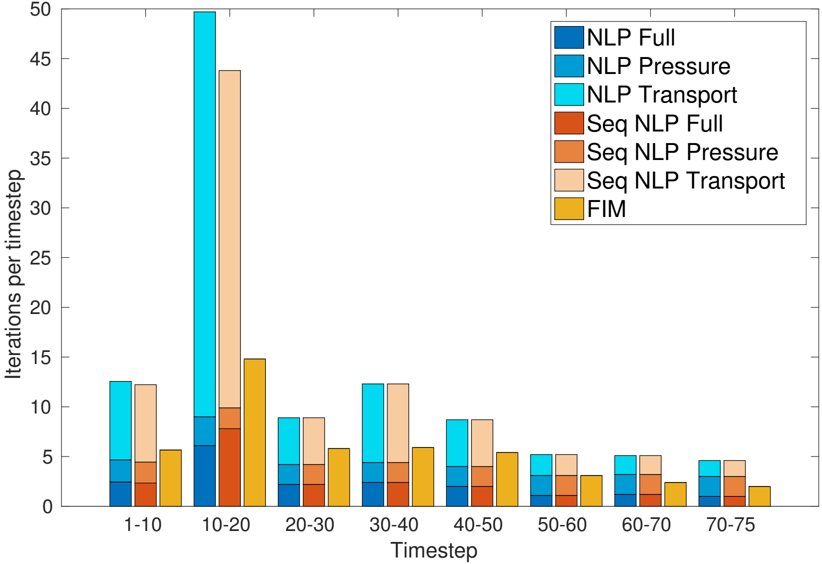
Water production rate

Recent experiments

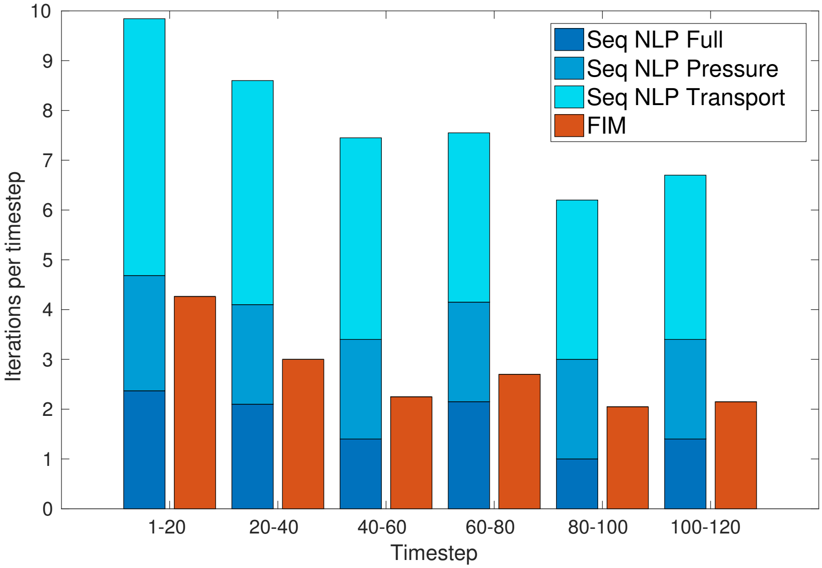
Using a splitting solver to

- improve iterates of fully implicit solver ("NLP"),
- generate initial value for fully implicit solver ("Seq NLP").

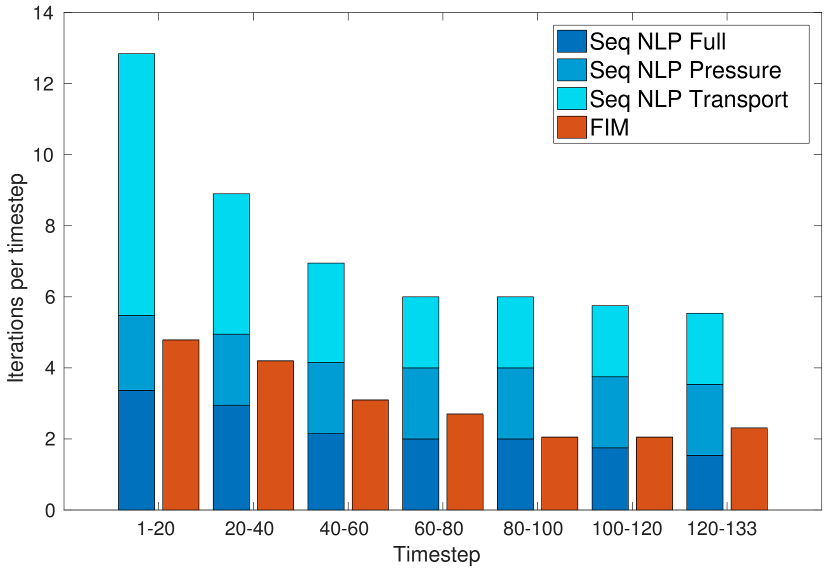
Recent experiments (SPE10 layer)



Recent experiments (SPE1)



Recent experiments (Simplified Norne)



Conclusion

- We can use sequential implicit methods on real-world field cases
  - ... at least for history-matching runs
- We can use reordering methods to solve the transport problem

Ongoing and future work

- Performance
  - Hope to make sequential implicit method roughly twice as fast as fully implicit.
  - Use transport solver as nonlinear preconditioner to improve performance of fully implicit simulation, possibly combined with CPR.
- Parallelization
  - Investigate possible approaches (multigrid-like, domain decomposition, etc.)
  - (Not in the near term)
- Robustness
  - Ensure successful runs for all available testcases (you can help!)
  - Investigate alternative solvers for single-cell problem (nested bracketed solver rather than Newton)

Availability

All simulators used are free and open-source.

- OPM website: opm-project.org
- OPM software sources: github.com/OPM

To run Norne as shown in this talk:

- fully implicit:
  `flow NORNE_ATW2013.DATA output_dir=fully-implicit`
  (also see Norne tutorial on opm website)
- reordering:
  `flow_reorder NORNE_ATW2013.DATA ds_max=0.1 output_dir=reorder`

`flow_reorder` is available as source in current master on GitHub, also as binary starting in upcoming release (2017.10)

Acknowledgements

Colleagues at SINTEF (ideas and discussions)

The OPM community (code and feedback)

Statoil (funding and data)

The CLIMIT program (funding)

The MSO4SC project (funding)

Thank you for listening!

Performance outlook

Fully implicit solver	
Stage	**Time (s)**
Assembly	321
Linear solver	299
Update	13

Reordering sequential solver	
Stage	**Time (s)**
Pressure solver	1380
Transport solver	345

Hoped-for potential	
Stage	**Time (s)**
Pressure solver	260
Transport solver	70

Rationale behind hoped-for potential:

- New pressure solver, does about half the assembly work (value and 1 derivative, rather than value and 3 derivatives) and linear solver deals with 1×1 rather than 3×3 blocks.
- Current transport solver does no convergence checking, brute-forces 5 global Gauss-Seidel iterations. Norne experiment leads us to expect average of close to 1 Gauss-Seidel iteration per cell.
- Current transport solver writes excessive log output, taking significant time.

Transport equation fluxes: fractional flow formulation

Establish upwind directions for each phase (following Brenier and Jaffré):

$$UP(\alpha, ij) \in i, j$$

(Function of p_β, ρ_β, λ_β for all β at both cells i and j; T_{ij} and v_T).

Phase fluxes are then given by:

$$(b_\alpha v_\alpha)_{ij} = b_{\alpha,ij} \frac{\lambda_{\alpha,ij}}{\sum_\beta \lambda_{\beta,ij}} (v_T + T_{ij} G_{ij})$$

where

$$\begin{aligned}
b_{\alpha,ij} &= b_{\alpha,UP(\alpha,ij)} \\
\lambda_{\alpha,ij} &= \lambda_{\alpha,UP(\alpha,ij)} \\
G_{\alpha,ij} &= \sum_{\beta \neq \alpha} \lambda_{\beta,ij} (D_{\alpha,ij} - D_{\beta,ij}) \\
D_{\alpha,ij} &= p_{o,j} - p_{o,i} - (p_{\alpha,j} - p_{\alpha,i} - g\rho_{\alpha,ij}(z_i - z_j))
\end{aligned}$$