Nonlinear solver techniques in reservoir simulation

InFoMM
Industrially Focused
Mathematical Modelling

Asbjørn Nilsen Riseth

University of Oxford

Supervisors: Patrick Farrell,
Tom Jönsthövel

A technical report for

InFoMM CDT Mini-Project 1

in partnership with

Schlumberger

Trinity 2015

Contents

1 Introduction

This report is the result of a 10 week mini-project collaboration between *Schlumberger Abingdon Technology Center* and *University of Oxford Mathematical Institute*. The objective was to investigate nonlinear solvers in reservoir simulation in practice, running experiments to test different approaches. Schlumberger develops two reservoir simulators used by major oil companies around the globe: ECLIPSE [16] and INTERSECT [18]. They contain massive code bases with sophisticated reservoir models, and it would not be feasible to implement and test new nonlinear techniques directly in the software for this project.

To enable experimentation with nonlinear solvers, we have therefore implemented a replica of the simplest black oil model available in *ECLIPSE E300*. This was still a tremendous task for a project of this length, and was successful thanks to the use of scientific toolkits Firedrake [13] and PETSc [4].

Solving nonlinear equations is a process consisting of several subcomponents. Figure 1 shows how much time is spent in the basec Newton solver's major components for a five hour INTERSECT simulation run. A choice of solver is effectively a choice of how much work to allocate to the different subcomponents. Such choices are problem specific, and should be guided by experience of the properties of the problem. This experience is effectively achieved through experimentation.

With our mini-simulator, a vast design space for composition of different nonlinear solvers is readily available to us via PETSc. The outcome of experiments show that we can reduce the number of nonlinear iterations required in our test models by 10%. This is achieved using the a nonlinear Krylov method NGMRES, preconditioned with the basic Newton method [19, 5].

The test cases considered have not been difficult enough to capture the hardest challenges experienced in industry. We have found that our test cases create nonlinear equations starting in the quadratic regime of Newton's method, which is incredibly difficult to improve upon. Noting that NGMRES already brings an improvement in these regimes is very promising. To further make use of our mini-simulator, we want to design test problems that cause problems for the basic Newton solver of the nature observed in practical applications.

Other techniques have also been tried. Quasi-Newton and preconditioner lagging have the potential to save time when employed in a sophisticated fashion. Applying these naively has not proved successful for the test cases considered. From our experiments we hypothesise that preconditioner lagging has potential to reduce timings if employed in a clever fashion.

The work from this report has resulted in several bug fixes and a feature contribution to the Firedrake and PETSc project. This has enabled preconditioner lagging on composite preconditioners, nonlinear solver composition of Newton and Quasi-Newton, and the improvement of NGMRES-accelerated Newton in stagnating regimes.

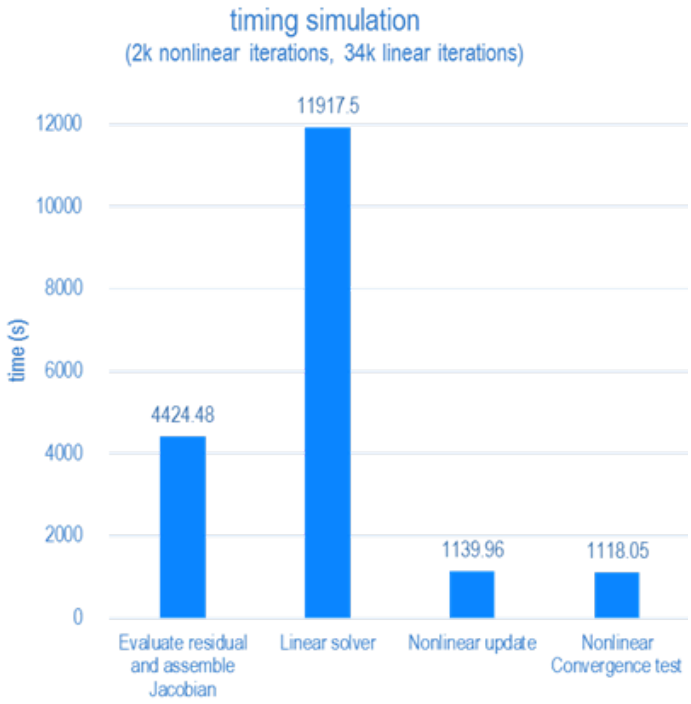


Figure 1: Example timing simulation from INTERSECT [18]. Novel nonlinear techniques can move some of the work done by the linear solver to other, more scalable components.

2 Governing equations

The equations solved by our reservoir model are from Schlumberger's ECLIPSE reservoir simulator [15, 16]. We consider the simplest case, a black oil model where no components can change phase. A derivation of the equations can be found in [10]. ECLIPSE solves the system using finite volume methods, whilst our formulation is written in the finite element framework. Using zero-order discontinuous Galerkin elements and the same numerical flux approximation, they result in the same numerical system. The equivalence of the two methods is described for the diffusion equation in appendix B.

2.1 Assumptions

We model the flow in the reservoir using three immiscible components: oil, gas and water. Each component corresponds to one phase in our fluid model; liquid, vapour and water respectively. In the oil industry this is sometimes referred to as a dead oil, dry gas-system. Capillary pressures are also omitted.

Let the reservoir be denoted as $\Omega \subset \mathbb{R}^3$ throughout. The primary variables of the system are component molar densities m_o, m_g, m_w of oil, gas and water, and pressure p. Their dimensions are $\frac{\text{mol}}{\text{L}^3}$ and $\frac{\text{M}}{\text{LT}^2}$. Together with the properties in table 1 we can fully describe the system. Expressions for properties depending on the primary variables are given in appendix A.

Property	**Symbol**	**Dependencies**	**Dimension**
Porosity	ϕ	p	-
Phase molar density	b_α	p	$\frac{\text{mol}}{\text{L}^3}$
Darcy velocity	v_α	m_α, p	$\frac{\text{L}}{\text{T}}$
External source/sink	σ_α	m_α, p	$\frac{\text{mol}}{\text{L}^3\text{T}}$
Absolute permeability tensor	K	-	L^2
Relative permeability	$k_{r\alpha}$	m_α, p	-
Dynamic viscosity	μ_α	p	$\frac{\text{M}}{\text{LT}}$
Mass density	ρ_α	p	$\frac{\text{M}}{\text{L}^3}$
Gravity	g	-	$\frac{\text{L}}{\text{T}^2}$
Saturation	S_α	m_α, p	-

Table 1: Overview of the reservoir properties of our model, including their dimension and dependency on pressure p and molar densities m_α. $\alpha = o, g, w$. Wells are modelled using σ_α.

The absolute permeability K is a diagonal matrix where the entries represent rock permeability along the three axes respectively.

2.2 Strong and weak formulation

The porous media flow model we use is derived from mass balance principles and a Darcy approximation for the flow. We work in a standard right-hand Cartesian coordinate frame, and set $\mathbf{g} = (0, 0, -g)^T$. Note that ECLIPSE uses a left-hand frame, where the positive z-direction is downward. For a given initial configuration in Ω, find p, m_α, $\alpha = g, o, w$ such that

$$\frac{\partial \phi m_\alpha}{\partial t} + \nabla \cdot (b_\alpha v_\alpha) = \sigma_\alpha \qquad \text{Mass balance} \qquad (1a)$$

$$v_\alpha = -K \frac{k_{r\alpha}}{\mu_\alpha} (\nabla p - \rho_\alpha \mathbf{g}) \qquad \text{Darcy flow} \qquad (1b)$$

$$\sum_\alpha S_\alpha = 1 \qquad \text{Volume constraint} \qquad (1c)$$

$$\frac{\partial b_\alpha v_\alpha}{\partial n} = 0 \qquad \text{No normal flow through } \partial\Omega. \qquad (1d)$$

One could use the volume constraint to eliminate one of the m_α's. Reservoir simulators approach this in different ways: *ECLIPSE E300* leaves it explicitly, whilst *INTERSECT* eliminates one primary variable [16, 17]. An argument for keeping all variables is that it gives the nonlinear solver a larger navigation space. One can take paths towards the root that leave the manifold defined by the constraint. As our simulator wants to replicate its behaviour, we have left the constraint in explicitly.

To solve the system, a finite volume discretisation is used. We formulate it in an equivalent weak sense, using piecewise constant discontinuous Galerkin elements ($\mathbb{P}^0_{DG}$). Background reading on the discontinuous Galerkin methods may be found in [14], including higher order examples for porous media flow.

We express the semi-discrete set of equations here. The resulting system of ODEs is solved using a fully implicit backward Euler scheme. For a given structured grid Ω_h of Ω, define Γ_{int} to be the set of interior facets. Define the mobility for a fluid to be $\lambda_\alpha = \frac{b_\alpha k_{r\alpha}}{\mu_\alpha}$. Let $\mathcal{V}_h = (\mathcal{V}_g, \mathcal{V}_o, \mathcal{V}_w, \mathcal{V}_p)$ be the $\mathbb{P}^0_{DG}$ space on Ω_h. Find $(m_g, m_o, m_w, p) \in \mathcal{V}_h$ such that

$$\int_\Omega \frac{\partial \phi m_\alpha}{\partial t} v \,\mathrm{d}x + \int_{\Gamma_{int}} \widetilde{K}\widetilde{\lambda_\alpha}\left(\widetilde{\nabla p} + \widetilde{\rho_\alpha}\mathbf{g}\right)(v^+ - v^-)\,\mathrm{d}s = \int_\Omega \sigma_\alpha v \,\mathrm{d}x \quad \forall v \in \mathcal{V}_\alpha, \alpha = g, o, w \tag{2a}$$

$$\int_\Omega \left(1 - \sum_\alpha S_\alpha\right) v \,\mathrm{d}x = 0 \qquad \forall v \in \mathcal{V}_p. \tag{2b}$$

The terms v^+, v^- refer to the limit value on a facet, taken from adjacent cells Ω^+, Ω^- respectively. Let $\hat{n}$ denote the positive unit normal for the facets. We approximate the flux across facets with an upwind Goudunov method, by setting

$$\widetilde{K} = \frac{1}{2}\left(\frac{K^+ + K^-}{K^+ \circ K^-}\right)\hat{n} \qquad \text{where } \circ \text{ is the Hadamard product,} \tag{3a}$$

$$\widetilde{\lambda_\alpha} = \begin{cases} \lambda_\alpha^- & \Phi_\alpha < 0 \\ \lambda_\alpha^+ & \Phi_\alpha \geq 0 \end{cases} \qquad \Phi_\alpha = \widetilde{\nabla p} + \widetilde{\rho_\alpha}\mathbf{g}, \tag{3b}$$

$$\widetilde{\nabla p} = \frac{p^+ - p^-}{\|h^+ - h^-\|} \qquad h \text{ denotes cell centres,} \tag{3c}$$

$$\widetilde{\rho_\alpha} = \frac{{S_\alpha}^+ \rho_\alpha^+ + {S_\alpha}^- \rho_\alpha^-}{{S_\alpha}^+ {S_\alpha}^-} \qquad \text{a saturation weighed average.} \tag{3d}$$

The permeability and mass density approximations can cause division by zero problems. For mass density, this is taken care of by taking the arithmetic average in the absence of a phase in two adjacent cells. For permeability, the face is considered impermeable, and $\widetilde{K} := 0$.

3 Implementation

A significant part of this project was spent setting up an environment to experiment with different approaches when solving the discretised equations. We call the implementation *Oildrake*. It is written in Python and built on top of *Firedrake* [13] and *PETSc* [4, 3, 9].

PETSc, the *Portable, Extensible Toolkit for Scientific Computation*, is used to solve the nonlinear system arising at each timestep. It makes experiments with the solvers easy: changing between sophisticated solvers may be done with command-line options. For example running a PETSc program with the arguments `-ksp_type gmres`

`-pc_type ilu` tells the software to solve a linear system using ILU(0)-preconditioned GMRES. Changing to a direct LU-solver can be done by passing `-ksp_type preonly -pc_type lu` instead. The options `-snes_type newtonls` and `-snes_type qn` tell PETSc to use basic Newton and Quasi-Newton nonlinear solvers respectively.

Firedrake is an automated system for solving partial differential equations using the finite element method. It is closely related to *the FEniCS project* [11], and uses some of the same components for turning a high-level mathematical finite-element language into efficient low-level code. It makes implementing the system of equations easy, as the code written strongly resembles eq. (2). The high-level mathematical formulation enables Firedrake to employ exact derivatives of the equations using automatic differentiation [1]. Firedrake was chosen over FEniCS because we needed support for hexahedra, which only Firedrake has at the time of writing.

3.1 Oildrake capabilities

We wish to experiment with nonlinear solvers, keeping other factors as close to *ECLIPSE E300* as possible. Notable capabilities and solver decisions are:

- Conforming rectangular cuboids in $\mathbb{R}^3$.
- FGMRES with the Constrained Pressure Residual (CPR) preconditioner [7].
- Fully implicit backward Euler time-discretisation.
- Adaptive heuristic timestepping based on the predicted saturation change.
- Anisotropic, heterogeneous permeability fields.
- Heterogeneous porosity fields.
- Changing the expressions of the properties in appendix A is easy, and the new Jacobian is automatically generated.
- Single-connected wells modelled as point-source/sinks.

The component molar densities must be positive, and to prevent unconstrained solvers from overshooting, we project any negative values into the feasible region after each nonlinear step.

Oildrake can only run in serial at the moment. Firedrake and PETSc support parallelism via MPI automatically, but this does not seem to work for Oildrake. A debugging session is necessary to find out why.

The convergence criteria used in Oildrake is different from Schlumberger's reservoir simulators. We terminate based on the ℓ_2-norms of the residuals, using the following default values:

- *Nonlinear solver*: Relative tolerance = 10^{-8}, absolute tolerance = 10^{-3}, maximum iterations = 20.

- *Linear solver*: Relative tolerance = 10^{-5}, absolute tolerance = 10^{-10}, maximum iterations = 20.

For E300, the convergence criteria is a custom max-norm on the residual the change in saturations. In practice this terminates the solver much earlier than Oildrake, i.e. Oildrake solves the equations to a tighter tolerance.

3.1.1 Validation against ECLIPSE

To ensure that the simulator works, we can compare it with ECLIPSE E300 on test models. Figure 3 shows a selection of cells in a model with permeability field derived from the SPE10 comparative solution project [8]. Both simulators predict the same flow patterns.

The initial pressure is set to 250 bar in the whole reservoir. We start the system in a highly unstable configuration, with three single phase layers water, oil and gas on top of each other. Due to mass density differences, the fluids flow due to gravity. Horizontal flow happens as a result of the heterogeneous permeability field shown in fig. 2.

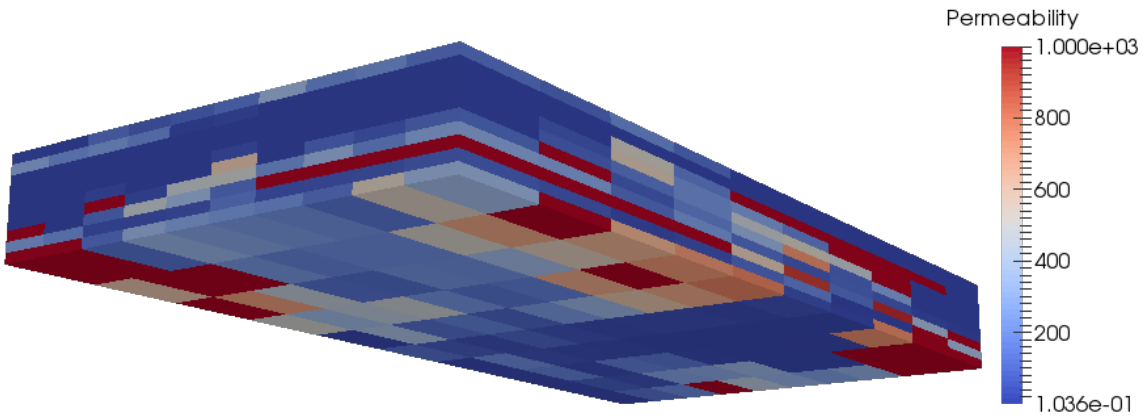


Figure 2: Permeability field for the SPE10Small model. It is highly varying, causing flow rates to be very different between cells.

3.2 Discrete system and linearisation

Oildrake solves the semi-discrete system in eq. (2) using a backward Euler discretisation. For a reservoir with N cells, this results in a nonlinear system $F_n : \mathbb{R}^{4N} \to \mathbb{R}^{4N}$ to be solved at each timestep.

The standard method to solve this system is Newton's method. Most of the taken by the nonlinear solver is spent finding the step direction given by the linearised system.

Figure 4 shows the sparsity pattern of the Jacobian arising from backward Euler. The different character of the pressure part of the system is the motivation behind the Constrained Pressure Residual (CPR) preconditioner [7, 16].

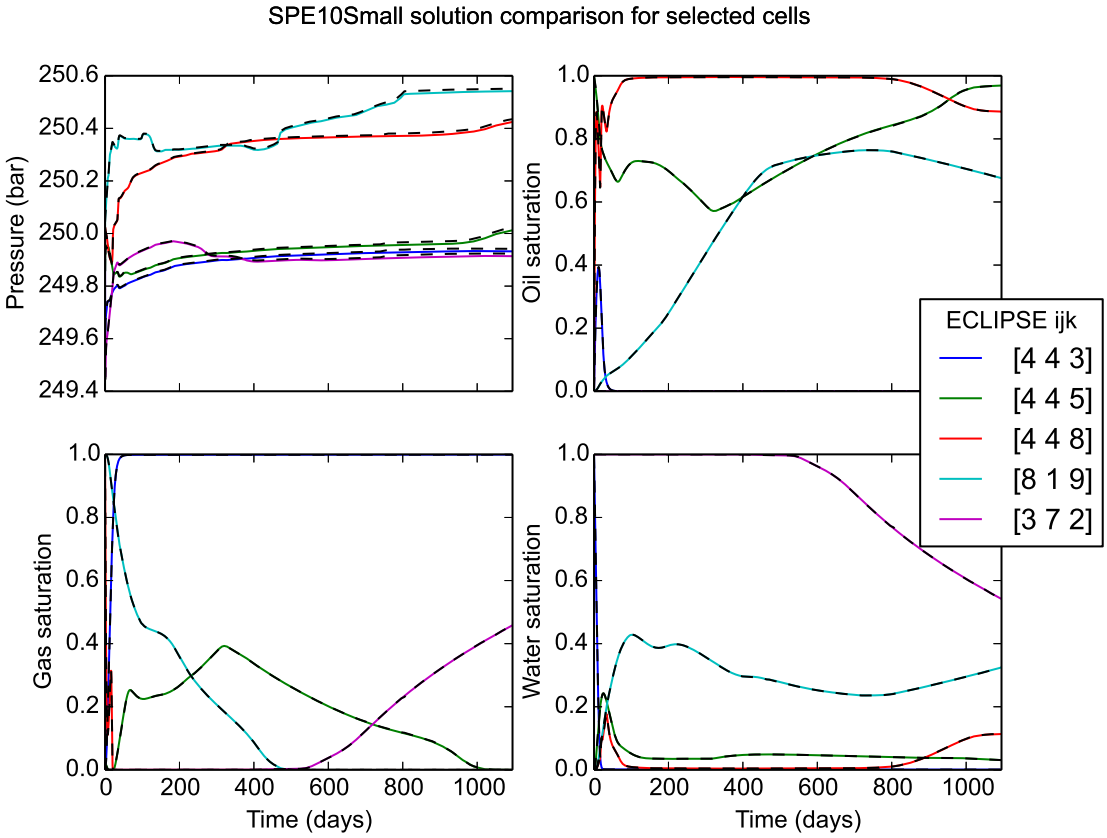


Figure 3: Gas inversion comparison between Oildrake and ECLIPSE, for select cells in the SPE10Small model. The black striped curves are the ECLIPSE solutions, which fit very well with our implementation. Note that the legend cells are given in ECLIPSE ijk-ordering, where the k's increases as we go down in the reservoir.

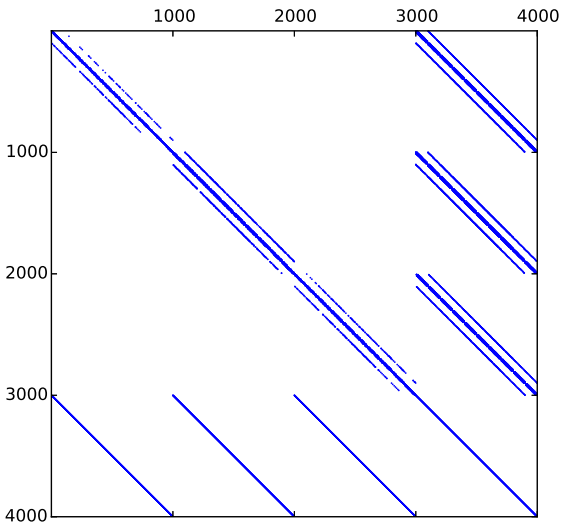


Figure 4: Sparsity pattern of the Jacobian from SPE10Small at time 0.1. The first 1000 rows/columns represent m_o, then comes m_g, m_w and p.

In reservoir simulation, FGMRES right preconditioned with CPR proves to be very effective. It is a composite multiplicative preconditioner. The first stage preconditioner is a Schur type approximation to the "pressure"-part of our equations eq. (1c). It applies one V-cycle of algebraic multigrid (AMG) to the approximate Schur complement, updating the pressure variables. The second stage uses a ILU(0) preconditioner on the whole system. Solving for the Newton step direction using FGMRES+CPR takes 5-10 iterations on average in our tests. All parts of the linear solver scale well in size and cores, apart from the AMG setup [2]. Reducing the AMG setup costs is important for practical problems, since they are increasingly solved on many-core systems.

For all our tests, we have used the Quasi-IMPES Schur approximation. In the preconditioning literature, this makes the inner CPR similar to a SIMPLE-type preconditioner [20].

4 Test problems

For this project, we set up a series of gas inversion tests. They all use cell discretisations where $\Delta x = 20, \Delta y = 10, \Delta z = 2$, representative for reservoir simulation cases used in practice.

The four tests used are

- SPE10Small: $10 \times 10 \times 10$ cells. Based on the SPE10 permeability field.
- Hetero: $24 \times 24 \times 24$. Smoothly varying permeability field, as shown in fig. 5
- Hetero25k: $20 \times 32 \times 40$. Smoothly varying permeability field.

- Hetero100k: $30 \times 50 \times 70$. Smoothly varying permeability field.

Starting the system with only gas in the bottom third, water in the top third and oil in the middle starts the flow immediately due to gravity and mass density differences.

The simulations are started with a timestep length of 0.1 days, and we use the same default settings as E300 for the adaptive timestepping with a saturation change target.

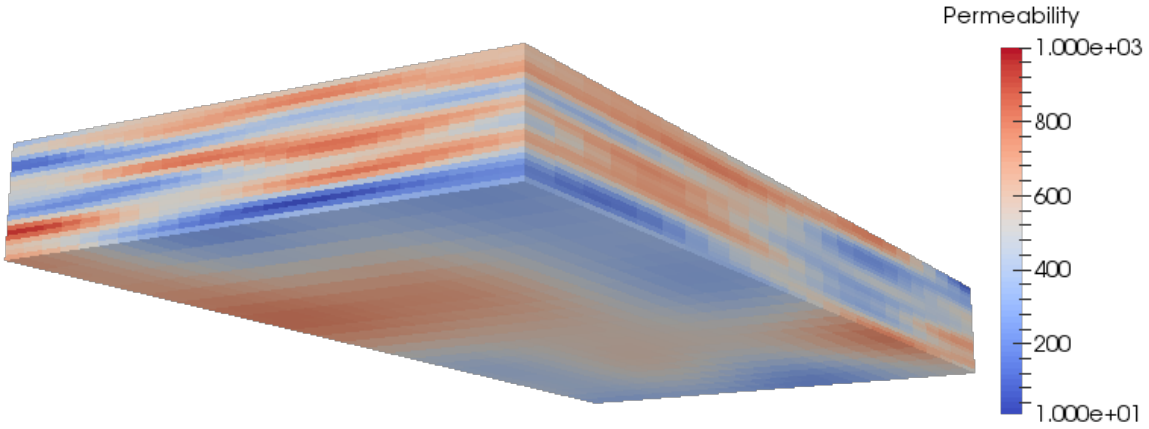
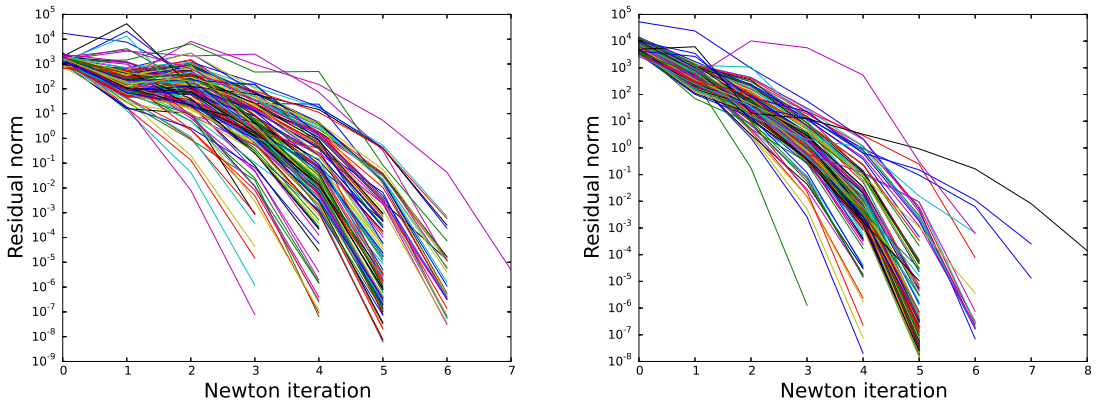


Figure 5: Permeability field for the Hetero model. The field is generated from a Fourier-like series with higher oscillation in the z-direction to simulate varying layers in a real reservoir.

Running preliminary tests with basic Newton indicates that the problems we have set up do not cause much trouble for the solver. Figure 6 show the residual norm vs number of Newton iterations for a selection of timesteps on the SPE10Small and Hetero models.



(a) SPE10Small model

(b) Hetero model

Figure 6: Running basic Newton on the test cases show that we lie in the quadratic regime for most timesteps. Beating Newton here is difficult, so setting up more difficult problems will be important in the future.

Table 2 shows the relative time taken from different components of a basic Newton solver. More than half the time in the linear solver is spent setting up the preconditioner. The test problems in this report are not very difficult for the linear solver, taking on average 5-7 linear steps per Newton iteration. For more difficult linear problems, the relative time spent in the linear solver is bound to increase against function evaluation, Jacobian assembly and preconditioner setup.

Operation	Share (%)
Linear solver	47
PC setup	29
Jacobian assembly	41
Function evaluation	10
Projection	2

Table 2: Share of time spent in different operations of a basic Newton solver. More than half the time of the linear solver is spent on the CPR preconditioner assembly. The numbers are taken from the Hetero100k model. The linear solver share decreases with the size of the problem.

5 Solver experiments

The intention of Oildrake is to easily test new solver techniques on reservoir models. With PETSc as a solver backend, this can be done setting options before running the simulation.

The efficiency of solvers is problem-specific. There is a given amount of work needed to be done, and different approaches basically move the work to other subcomponents. For example, lagging the preconditioner may decrease the number of preconditioner setups, at the expense of running more Krylov iterations.

In this section, we show the result of two approaches: preconditioner lagging and NGMRES with Newton as a preconditioner. The subsections show what options tell PETSc to use the solvers. In addition, any linear solver settings must be passed to the Newton component.

Lagging the preconditioner is an example of a tactic of reducing accuracy of single iterations to see if what one can get away with. NGMRES on the other hand is used a nonlinear Krylov method used to accelerate each Newton step, making each step more efficient.

The tests on Oildrake show that NGMRES with nonlinear preconditioning performs better than basic Newton, whilst naive preconditioner lagging does worse for the test cases considered.

A Quasi-Newton (QN) solver was also tested as part of this project. The idea behind it is to trade more nonlinear iterations for solving the same matrix system. QN assembles the Jacobian at a given step, and then does rank-one updates to approximate the linear system in consecutive steps. For problems where basic Newton

is stagnant over several iterations, QN can replace these expensive steps with much cheaper ones. Using nonlinear solver composition, we can use QN to take us cheaply closer to the correct solution, before falling back to Newton with its great convergence properties. In the test cases considered, basic Newton is already in the quadratic regime, so this trade did not pay off.

5.1 Application specific timings

In interpreting the results from Oildrake, we stress that the relative runtime of different subcomponents of the solvers can't be mapped to performance in ECLIPSE or INTERSECT. The most important results to take from these runs are the number of calls to different subcomponents of the solvers. These numbers can give an indication of how the new approaches will do in the industry simulators.

The two main differences to ECLIPSE and INTERSECT are

1. *Parallelism*: Industry simulators are run in parallel, but these tests have been done in serial. Some components scale well in parallel, others don't. For large problems, we want to shift work from nonscalable to scalable subproblems.

2. *Evaluation and assembly*: Oildrake's Jacobian assembly and function evaluation has not been optimised to the same degree as industry simulators.

The run time per operation varies between runs in Oildrake. Caching and other activity on the computer are two causes for this.

5.2 NGMRES preconditioned with basic Newton

Petsc options: `-snes_type ngmres -snes_ngmres_restart_fm_rise -snes_npc_side right -npc_snes_type newtonls`

Basic Newton forgets its iteration history, and will only use information at the current step to decide where to move forward. Nonlinear GMRES will also use the insight of previous iterations to accelerate convergence. It is suggested in [19] that NGMRES preconditioned with an application specific nonlinear solver can improve convergence.

Algorithm 1 describes the NGMRES algorithm as it is implemented in PETSc. We won't go in to the sufficiency requirements in the iteration update, but refer to the PETSc documentation.

Preliminary tests with NGMRES showed stagnation for certain timesteps. The cause was that the Newton preconditioner wanted to step in a direction that significantly increased the residual, while NGMRES hindered this step. A Newton step increasing the residual may be an attempt to move out of a stagnation area, from which convergence can go more rapidly.

This was dealt with by resetting the Krylov subspace after such an attempt, which improved the performance of the algorithm for our test problems. One can tell PETSc to do this by passing the option `-snes_ngmres_restart_fm_rise`. The feature was

Algorithm 1 Right-preconditioned NGMRES

procedure NGMRES($F, x_i, \ldots, x_{i-m+1}$)

$x_i^M = M(x_{i-1})$ ▷ M is the preconditioner

$F_i^M = F(x_i^M)$

minimize $\left\| F\left(\left(1 - \sum_{k=i-m}^{i-1} \alpha_i\right) x_i^M + \sum_{k=i-m}^{i-1} \alpha_k x_k\right)\right\|_2$ over $\{\alpha_{i-m}, \ldots \alpha_{i-1}\}$

$x_i^A = \left(1 - \sum_{k=i-m}^{i-1} \alpha_i\right) x_i^M + \sum_{k=i-m}^{i-1} \alpha_k x_k$

$x_{i+1} = x_i^A$ or x_i^M if x_i^A is insufficient.

end procedure

contributed as part of this mini-project, and should be available in releases after version 3.6.0.

Tables 3 and 4 show that accelerating the Newton steps with NGMRES indeed reduces the number of nonlinear solves needed per timestep. The main cost is extra function evaluations, but overall the time savings are greater in our tests.

	Nonlinear solves		**Linear solves**		**Residual evaluations**	
Test	Newton	NGMRES	Newton	NGMRES	Newton	NGMRES
SPE10	854	769	6030	5278	1037	1721
Hetero	1113	991	6333	5534	1349	2218
Hetero25k	1719	1531	10000	8642	2095	3438
Hetero100k	2740	2479	15906	14003	3317	5535

Table 3: Comparison of number of operations using basic Newton and NGMRES. NGMRES reduces the number of nonlinear iterations by about 10%, at the expense of increasing the number of residual evaluations by 60%-70%. There is one preconditioner assembly per nonlinear solve.

Test	Total time	Linear	PC Setup	Residual	Jacobian
SPE10 Basic	73.2	14.1	6.5	14.5	41.6
SPE10 NGMRES	75.6	11.5	5.5	21.6	36.2
Hetero Basic	457.9	180.6	108.6	57.8	210.4
Hetero Hetero25k	465.9	166.1	97.6	93.4	187.7
Hetero25k Basic	1282.4	552.4	331.7	145.8	558.8
Hetero25k NGMRES	1253.4	490.5	297.2	235.6	496.4
Hetero100k Basic	8832.5	4179.3	2524.1	866.9	3610.8
Hetero100k NGMRES	8554.7	3643.0	2218.9	1410.6	3312.0

Table 4: Run time comparison of different parts of the nonlinear solvers in seconds. NGMRES reduces runtime for large problems. As the problems get larger, the savings from reducing the number of nonlinear iterations needed are greater than the cost of extra function evaluations.

5.3 Preconditioner lagging

PETSc option: `-snes_preconditioner_lagging <number>`

The preconditioner setup is costly and does not scale as well as other solver components. This motivates the attempt to set up the preconditioner, then reuse it for later linear systems.

In our tests, lagging the preconditioner significantly reduces the number of PC set ups. Tables 5 and 6 show the results of setting up the preconditioner every second and third nonlinear iteration. The extra number of linears used is less than twice the reduction in number of PC set ups. Running in serial with Oildrake, the extra cost of linear and nonlinear steps was larger than the savings in PC set ups.

The lagged preconditioner usually performed better in later Newton iterations. We hypothesise that the step lengths are shorter at these iterations, so the lagged preconditioner approximates the current step better. Making preconditioner lagging decisions based on the length of the previous nonlinear step may preserve sufficient accuracy. We did not have time to test this in Oildrake.

The first lagged preconditioner can often take more than 100 iterations. In our tests, we capped them at 20 per Newton step. As a result, the number of nonlinears needed per step has increased, as can be seen in table 5.

Test	Nonlinears	Linears	PC Setups	Residuals
SPE10 Basic	854	6030	854	1037
SPE10 Lag 2	863	7604	479	1046
SPE10 Lag 3	868	9158	354	1051
Hetero Basic	1113	6333	1113	1349
Hetero Lag 2	1159	11066	660	1395
Hetero Lag 3	1282	15388	508	1518
Hetero25k Basic	1719	10000	1719	2095
Hetero25k Lag 2	1804	16895	1028	2180
Hetero25k Lag 3	2016	23666	811	2392
Hetero100k Basic	2740	15906	2740	3317
Hetero100k Lag 2	2910	28571	1626	3488
Hetero100k Lag 3	3367	41910	1364	3945

Table 5: Comparison of number of operations using basic Newton and preconditioner lagging. Lagging the preconditioner once decreases the PC setups by about 40%. In the SPE10 case, it increases the number of linears by 26%. In the Hetero-tests, the linears required increased by 70%-80%.

Test	Total time	Linear	PC Setup	Residual	Jacobian
SPE10 Basic	73.7	14.6	6.5	14.4	41.9
SPE10 Lag 2	73.9	13.3	3.8	14.7	42.7
SPE10 Lag 3	73.2	13.6	2.8	14.6	42.3
Hetero Basic	476.5	188.2	113.0	58.0	220.5
Hetero Lag 2	492.4	197.2	66.3	60.1	224.9
Hetero Lag 3	583.5	246.7	53.4	66.6	258.4
Hetero25k Basic	1385.6	589.4	350.5	147.2	617.3
Hetero25k Lag 2	1454.3	620.5	210.7	153.0	653.7
Hetero25k Lag 3	1585.6	707.5	159.8	168.6	677.2
Hetero100k Basic	8894.0	4109.0	2488.7	834.4	3774.2
Hetero100k Lag 2	9740.6	4608.7	1504.0	886.6	4066.4
Hetero100k Lag 3	11509.1	5621.6	1236.8	999.1	4687.5

Table 6: Run time comparison of different parts of the nonlinear solvers in seconds. The relative PC setup costs on small problems in serial are never large enough to justify the additional linear iterations needed.

6 Summary

Investigations into nonlinear solvers for reservoir simulation has been done using our mini-simulator Oildrake. It is currently replicating the simpler settings of ECLIPSE E300, and is easily extensible. The software gives the user easy access to novel mathematical techniques for solving nonlinear equations via PETSc, which can guide decisions for further study and implementation in industry applications.

Accelerating the nonlinear Newton solver with NGMRES reduces the number of nonlinear iterations by 10%. The tests considered in the report are shown to lie in the quadratic regime of Newton's method. That NGMRES can still improve the convergence in these regimes is very encouraging for further study on harder problems. The next step following up on this project should be to test cases with wells, and to introduce phase changes to the hydrocarbons.

The CPR preconditioner works very well on the linearised systems we have considered, but the setup phase is costly and does not scale as well as the other components. The number of setups can be reduced by lagging the preconditioner. From our experiments, we hypothesise that lagging decisions made based on previous nonlinear step lengths can save setup time without reducing its effect much.

In addition to designing more difficult test cases, there are four areas that deserve consideration.

1. MPI is supposed to work out of the box with Firedrake. This is not the case for Oildrake, so the toolchain must be debugged.

2. The convergence criteria used in the industry solvers do not follow the PETSc residual tolerance model. We should rethink our approach to make it more similar to ECLIPSE or INTERSECT.

3. Nonlinear instabilities, for example around wells, can be better dealt with in a localised manner. A domain decomposition solver that can do this is Nonlinear Additive Schwarz (NASM) [6]. PETSc supports this solver, and Firedrake support will be implemented soon, giving us the opportunity to test NASM on reservoir models in Oildrake.

4. Phases appearing and disappearing in cells activates and deactivates parts of the equations in the Schlumberger's simulators. This is a question of how to deal with variables at constraint boundaries, which may be approached as a complementarity problem [12].

A Properties and input values

The input values used for the experiments in this report are given in the same units as ECLIPSE's "metric" setting. These cause the units of the accumulation and flux terms to differ, so the flux term must be multiplied by a conversion factor. The factor is called a Darcy constant, and is defined as $C_{Darcy} = 8.52702 \times 10^{-3}\text{cPm}^2/\text{day/bar}$ [15]. We note that "day" is not an SI unit.

Symbol	Value	$\times 10^{\text{x}}$	Units
ρ_{sc}^o	8.0	2	kg/m^3
ρ_{sc}^g	9.907	-1	kg/m^3
ρ_{sc}^w	1.022	3	kg/m^3
M^o	1.20	2	kg/kmol
M^g	2.5	1	kg/kmol
M^w	1.8025	1	kg/kmol
g	9.80665	-5	$\text{m}^2\text{kg/bar}$

(a) Physical constants.

Symbol	Value	$\times 10^{\text{x}}$	Units
p_{ref}	2.5	2	bar
B_{pref}^w	1.03	0	-
μ_{ref}^w	3.0	-1	cP
C_w	0.0	0	1/bar
C	4.1	-5	1/bar
C_{rock}	5.3	-5	1/bar

(b) Reservoir specific input values.

Table 7: Input values used when calculating the value of the properties listed in this section. Water is considered incompressible.

A.1 Properties

The expressions used for the properties of the fluids from table 1 are listed here. Our experiments have been run with the constant values in table 7. For further explanation of the physical meaning of the following properties, see [10].

B_α is the *formation volume factor*, and is used to calculate phase molar density and mass density.

$$B_\alpha \text{ is given by interpolation using table 8} \qquad \alpha = o, g \tag{4}$$

$$B^w = \frac{B^w_{pref}}{1 + X + \frac{X^2}{2}} \qquad X = C \cdot (p - p_{ref}) \tag{5}$$

$$b_\alpha = \frac{b^\alpha_{sc}}{B_\alpha} \qquad b^\alpha_{sc} = \frac{\rho^\alpha_{sc}}{M_\alpha} \tag{6}$$

$$S_\alpha = \frac{m_\alpha}{b_\alpha} \tag{7}$$

$$\mu_\alpha \text{ is given by interpolation using table 8} \qquad \alpha = o, g \tag{8}$$

$$\mu^w = \frac{\mu^w_{pref}}{1 + X_v + \frac{X_v^2}{2}} \qquad X_v = C_v \cdot (p - p_{ref}) \tag{9}$$

$$\rho_\alpha = \frac{\rho^\alpha_{sc}}{B_\alpha} \qquad \text{in cells} \tag{10}$$

$$\rho_{\alpha,ij} = \frac{S_{\alpha,i}\rho_{\alpha,i} + S_{\alpha,j}\rho_{\alpha,j}}{S_{\alpha,i} + S_{\alpha,j}} \qquad \text{on facets} \tag{11}$$

$$k_{r\alpha} = S_\alpha \tag{12}$$

$$\phi = \varphi \cdot \left(1 + X_r + \frac{X_r^2}{2}\right) \qquad X_r = C_{rock} \cdot (p - p_{ref}) \tag{13}$$

The mass densities on facets are calculated using a saturation weighted average. One needs to be careful if the saturation in both adjacent cells is zero. In this case, take $\rho_{\alpha,ij} = \frac{\rho_{\alpha,i}+\rho_{\alpha,j}}{2}$.

Setting relative permeabilities equal to the saturations is an oversimplification. Better approximations use a quadratic dependence, possibly with a "critical saturation" where $k_{r\alpha}$ is zero for saturations smaller than this value.

p	$B^o(p)$	$\mu^o(p)$
50	1.18	0.8
600	1.08	1.6

p	$B^g(p)$	$\mu^g(p)$
50	2.05×10^{-2}	1.4×10^{-2}
600	3.9×10^{-3}	2.5×10^{-2}

Table 8: Oil and gas interpolation tables used to calculate formation volume factors and viscosities.

B Finite Volumes as a Discontinuous Galerkin scheme

Schlumberger discretises their reservoir models using finite volumes. In this report we use a discontinuous Galerkin finite element formulation so that Firedrake[13] and the UFL language [1] may be employed. The piecewise constant DG formulation is equivalent to the finite volume approach. To see this, consider the following PDE on $\Omega \subset \mathbb{R}^n$ with homogeneous Neumann BCs:

$$\frac{\partial u}{\partial t} - \Delta u = 0. \tag{14}$$

For the following, let $\{\Omega_i \mid i \in \mathcal{I}\}$ be a partition creating a structured grid of Ω, for some index set $\mathcal{I}$. Define $\tau_{ij} = \Omega_i \cap \Omega_j$, and let Γ_{int} denote the union of all the interior facets. $\mathcal{N}(i)$ is the set of indices j so that $|\tau_{ij}| > 0$. Let $\mathcal{V}_h$ be a $\mathbb{P}^0_{DG}$ space with basis $\{\phi_i = \mathbf{1}_{\Omega_i} \mid i \in \mathcal{I}\}$.

Our $\mathbb{P}^0_{DG}$ approximation $u = \sum_{i\in\mathcal{I}} u_i\phi_i$ of eq. (14) satisfies

$$\int_\Omega \frac{\partial u}{\partial t}\, v \,\mathrm{d}x + \int_{\Gamma_{int}} \frac{u^+ - u^-}{\|h^+ - h^-\|}(v^+ - v^-)\,\mathrm{d}s = 0 \qquad \forall v \in \mathcal{V}_h. \tag{15}$$

For a given ordering of $\mathcal{I}$, the notation u^+ and u^- is used denote the limit value of u taken from the two cells that share a given facet facet. h denotes the center points of the cells.

The $\mathbb{P}^0_{DG}$ formulation comes from the cell-wise weak form of eq. (14). u must satisfy

$$\int_{\Omega_i} \frac{\partial u}{\partial t} v \,\mathrm{d}x + \int_{\Omega_i} \nabla u \cdot \nabla v \,\mathrm{d}x - \int_{\partial\Omega_i} v \frac{\partial u}{\partial n}\,\mathrm{d}s = 0 \qquad \forall i \in \mathcal{I}, v \in \mathcal{V}_h. \tag{16}$$

On the interior facets the flux is not defined, so we choose a flux approximation and set

$$\int_{\partial\Omega_i} v \frac{\partial u}{\partial n}\,\mathrm{d}s := \sum_{j\in\mathcal{N}(i)} \int_{\tau_{ij}} v|_i \frac{u|_j - u|_i}{\|h|_j - h|_i\|}\,\mathrm{d}s. \tag{17}$$

For our $\mathbb{P}^0_{DG}$ space $\mathcal{V}_h$, v is piecewise constant and $\nabla v = 0$ in each cell. Hence, eq. (16) becomes

$$\int_{\Omega_i} \frac{\partial u}{\partial t} v \,\mathrm{d}x - \sum_{j\in\mathcal{N}(i)} \int_{\tau_{ij}} v_i \frac{u_j - u_i}{\|h_j - h_i\|}\,\mathrm{d}s = 0 \qquad \forall i \in \mathcal{I}, v \in \mathcal{V}_h. \tag{18}$$

We now sum over $\mathcal{I}$ in eq. (18), and note that each interior facet is visited twice. With the "+","-" style ordering of $\mathcal{I}$, this yields the $\mathbb{P}^0_{DG}$ formulation eq. (17).

If we instead consider eq. (18) in its own right, we get

$$\frac{\partial u_i}{\partial t}|\Omega_i| - \sum_{j\in\mathcal{N}(i)} \frac{u_j - u_i}{\|h_j - h_i\|}|\tau_{ij}| = 0 \qquad \forall i \in \mathcal{I}. \tag{19}$$

This is exactly the finite volume formulation of eq. (14) with the flux approximation $\frac{\partial u}{\partial n}\big|_{\tau_{ij}} = \frac{u_j - u_i}{h_{ij}}$.

References

[1] M. S. Alnæs, A. Logg, K. B. Ølgaard, M. E. Rognes, and G. N. Wells. Unified form language: A domain-specific language for weak formulations of partial differential equations. *ACM Transactions on Mathematical Software*, 40(2):9:1–9:37, Mar. 2014. doi: 10.1145/2566630.

[2] A. H. Baker, R. D. Falgout, T. V. Kolev, and U. M. Yang. Scaling hypre's multigrid solvers to 100,000 cores. In *High-Performance Scientific Computing*, pages 261–279. Springer, 2012.

[3] S. Balay, W. D. Gropp, L. C. McInnes, and B. F. Smith. Efficient management of parallelism in object oriented numerical software libraries. In E. Arge, A. M. Bruaset, and H. P. Langtangen, editors, *Modern Software Tools in Scientific Computing*, pages 163–202. Birkhäuser Press, 1997.

[4] S. Balay, S. Abhyankar, M. F. Adams, J. Brown, P. Brune, K. Buschelman, L. Dalcin, V. Eijkhout, W. D. Gropp, D. Kaushik, M. G. Knepley, L. C. McInnes, K. Rupp, B. F. Smith, S. Zampini, and H. Zhang. PETSc users manual. Technical Report ANL-95/11 - Revision 3.6, Argonne National Laboratory, 2015.

[5] P. Brune, M. Knepley, B. Smith, and X. Tu. Composing scalable nonlinear algebraic solvers. *Argonne National Laboratory, Preprint ANL/MCS-P2010-0112*, 2013.

[6] X.-C. Cai, D. E. Keyes, and L. Marcinkowski. Non-linear additive schwarz preconditioners and application in computational fluid dynamics. *International journal for numerical methods in fluids*, 40(12):1463–1470, 2002.

[7] H. Cao, H. A. Tchelepi, J. R. Wallis, H. E. Yardumian, et al. Parallel scalable unstructured CPR-type linear solver for reservoir simulation. In *SPE Annual Technical Conference and Exhibition*. Society of Petroleum Engineers, 2005.

[8] M. Christie, M. Blunt, et al. Tenth SPE comparative solution project: A comparison of upscaling techniques. In *SPE Reservoir Simulation Symposium*. Society of Petroleum Engineers, 2001.

[9] L. D. Dalcin, R. R. Paz, P. A. Kler, and A. Cosimo. Parallel distributed computing using Python. *Advances in Water Resources*, 34(9):1124–1139, 2011. New Computational Methods and Software Tools.

[10] M. l. C. Christensen and K. L. Eskildsen. Nonlinear multigrid for efficient reservoir simulation. Master's thesis, Technical University of Denmark, Department of Applied Mathematics and Computer Science, 2012. URL `http://www2.imm.dtu.dk/pubdb/views/publication_details.php?id=6687`.

[11] A. Logg, K.-A. Mardal, and G. Wells. *Automated solution of differential equations by the finite element method: The FEniCS book*, volume 84. Springer, 2012.

[12] T. S. Munson. *Algorithms and environments for complementarity*. PhD thesis, University of Wisconsin–Madison, 2000.

[13] F. Rathgeber, D. A. Ham, L. Mitchell, M. Lange, F. Luporini, A. T. McRae, G.-T. Bercea, G. R. Markall, and P. H. Kelly. Firedrake: automating the finite element method by composing abstractions. *Submitted to ACM Transactions on Mathematical Software*, 2015.

[14] B. Rivière. *Discontinuous Galerkin Methods for Solving Elliptic and Parabolic Equations*. Society for Industrial and Applied Mathematics, 2008.

[15] *ECLIPSE: Reference Manual 2015.1.* Schlumberger, 2015.

[16] *ECLIPSE: Technical Description 2015.1.* Schlumberger, 2015.

[17] *INTERSECT: Technical Description 2015.1.* Schlumberger, 2015.

[18] *INTERSECT: User Guide 2015.1.* Schlumberger, 2015.

[19] H. d. Sterck. Steepest descent preconditioning for nonlinear GMRES optimization. *Numerical Linear Algebra with Applications*, 20(3), 2013.

[20] M. ur Rehman, C. Vuik, and G. Segal. Block preconditioners for the incompressible Stokes problem. In *Large-Scale Scientific Computing*, pages 829–836. Springer, 2010.