A Reservoir Simulator Prototype: Multi-phase Software Abstraction and Multiple Solvers

Musa Maharramov (musa@sep.stanford.edu)

Stanford Exploration Project

March 31, 2015

Contents

Abstract

This report expands on earlier reports to provide a detailed description of reservoir and production/injection well simulation experiments by the version 2.0 of `OOMP_RS` Object-Oriented Multi-Solver reservoir simulator. I describe the polymorphic multi-phase simulation framework implemented in the simulator software, and pay special attention to how the proposed framework obviates the need for customizing the back-end solver for specific modeling cases (i.e., two-phase or black oil) or alternative finite-difference solvers.

1 Overview

We begin with a detailed statement of the governing partial differential equations and boundary conditions. In this report we include production/injection wells in the governing equations, allowing arbitrary time-dependent well controls of flow-rates and (maximum/minimum) bottom-hole pressures. Albeit the specific two-phase simulator implementation in `OOMP_RS` provides for *single-completion* vertical wells, we discuss multiple-completion wells and their impact on the FIM/IMPES Jacobian and residual computation.

Once we have formulated a closed system of equations, we list quantities and values that are specified as inputs – e.g., the capillary pressure function, viscosity, fluid and rock compressibilities and densities. Once all of the known and unknown functions have been listed, we formulate a *boundary-value problem* for the system of governing equations that describes our reservoir system.

Our next step will be to introduce physical units for all of the key dimensional quantities involved in the simulation[1], and the associated conversion factors, if applicable. This is done specifically for the purpose of enabling a formulation of discrete analogues of the governing partial differential equations that would allow us to simulate the evolution of reservoir parameters in *reservoir* or *customary* units.

Having exhaustively identified all the unknowns, coefficients and conversion factors associated with our PDEs, we proceed to *discretize* the equations for the purpose of approximating them on a *finite-volume* grid. Differential equations lend themselves to multiple discretizations into *finite-difference equations*, depending on the desired accuracy and stability characteristics and employed approximation techniques. This naturally leads to the discussion of *two principal* discretization methods and the associated numeric algorithms for solving discrete boundary-value problems for the obtained finite-difference equations.

[1] except for the quantities that have the "dimension" of *fraction* or *part*

With the algorithms and underlying mathematical techniques defined, we proceed to describing the *data structures* that we use in the second version of our `OOMP_RS` reservoir simulator to store and manipulate all of the known and unknown quantities arising in the solution of the reservoir simulation finite-difference problems.

A separate section is dedicated to the numerical study of the two reference and one test cases provided as part of the Phase 3b problem description. A special driver program has been developed specifically for this task, that demonstrates the two key modules of `OOMP_RS` – `multiphase.F90` and `twophase.F90` – on these examples. In addition to providing computed residuals and the Jacobian for each of the reference and test cases, we demonstrate stability and convergence of both FIM and IMPES implementations of our reservoir simulator.

We conclude the report by providing *code samples* for the algorithms described earlier in the report, demonstrating all the key steps and interaction with the data structures. We describe key *object-oriented coding* techniques that allow a straightforward extension of our approach to implement alternative reservoir models – e.g., the *black oil* or even general *multicomponent* models – and numerical algorithms – e.g. using GMRES (see [10]) instead of the *conjugate residuals* used in the current version of `OOMP_RS`.

This report adheres to the object-oriented `Fortran 2003` notation (see [8]) as this is the language used in our reservoir simulator.

2 Governing Equations

Our objective is to develop a reservoir simulator for modeling a two-component two-phase oil-water system. In our treatment of the governing equations we generally follow [2], citing any deviations as necessary. We consider the two phases as *immiscible* (see [2]), assuming that each component can exist only in its primary phase. This results in a significant simplification of the resulting equations while still allowing for upgrading to a fully-fledged *black-oil* model at a later time. The relative ease of such an upgrade is due to the fact that in the black-oil system only the gas component is allowed to be present in other phases, often assuming no gas dissolved in the water phase.

Since the requirement of this phase is to design a reservoir simulator for a vertical 2D reservoir, we will include gravity terms in the formulation[2].

The oil conservation equation for a two-phase two-component oil-water system with immiscible components is

$$\begin{aligned} &\frac{\partial}{\partial x}\left[k\frac{\lambda_o}{B_o}\left(\frac{\partial P_o}{\partial x}-\gamma_o\frac{\partial D}{\partial x}\right)\right] + \\ &\frac{\partial}{\partial z}\left[k\frac{\lambda_o}{B_o}\left(\frac{\partial P_o}{\partial z}-\gamma_o\frac{\partial D}{\partial z}\right)\right] = \frac{\partial}{\partial t}\left(\phi\frac{S_o}{B_o}\right)+\tilde{q}_o, \end{aligned} \tag{1}$$

[2]Note that both the extensible multiphase framework (`multiphase.F90`) and the two-phase implementation (`twophase.F90`) in `OOMP_RS` have been been developed for tilted 3D grids

where $\lambda = \frac{k_{ro}}{\mu_o}$ is the *mobility*, μ_o – *viscosity*, B_o – *formation volume factor* and k_{ro} – *relative permeability* for the oil component. The coefficient k is the *geometric rock permeability* (independent of fluid),ϕ is the *rock porosity* and γ_o is the *gravitational constant* for oil. The unknown quantities in equation (1) are the *oil phase pressure* $P_o = P_o(x, z, t)$ and *oil component saturation* $S_o = S_o(x, z, t)$ – all the other quantities are specified as either constants or functions of the unknowns. The well production/injection term $\tilde{q}_o$ in this report is a function of *state variables*, we will relate this term to other known and *unknown* quantities through *well modeling* equations. Function D in (1) is the *true vertical depth* (TVD) of the subsurface point (x, z). This is required to model *tilted* reservoir grids where z may not represent the TVD. Water conservation equations are formulated in an analogous way:

$$\begin{aligned} &\frac{\partial}{\partial x}\left[k\frac{\lambda_w}{B_w}\left(\frac{\partial P_w}{\partial x} - \gamma_w \frac{\partial D}{\partial x}\right)\right] + \\ &\frac{\partial}{\partial z}\left[k\frac{\lambda_w}{B_w}\left(\frac{\partial P_w}{\partial z} - \gamma_w \frac{\partial D}{\partial z}\right)\right] = \frac{\partial}{\partial t}\left(\phi\frac{S_w}{B_w}\right) + \tilde{q}_w, \end{aligned} \tag{2}$$

where the subscript "w" denotes the corresponding parameter for the water component or phase. We have two conservation equations (1) and (2) and seek *four* unknowns – oil and water pressures and saturations. We complement the mass balance equations with the *capillary pressure relationship*:

$$P_c(S_w) = P_o - P_w, \tag{3}$$

where the difference between the pressures of the oil and water phases (capillary pressure) is assumed to be a known[3] *function of the water component saturation*. Note that in case we modelled more than two components/phases, capillary-pressure relations would be required for *each pair* of phase pressures, that might be, in principle, functions of multiple saturations. In our two-phase model, however, we simply assume that a capillary pressure is specified as a *tabulated* or *power function of water saturation*.

The final equation that allows us to close the system of governing equations relates the component saturations and is the obvious requirement that the saturations add up to 1:

$$S_o + S_w = 1. \tag{4}$$

Now, equations (1,2,3,4) form a closed system of four equations for four unknowns. Since (3) and (4) are extremely simple, we can analytically reduce the number of unknowns to two. For our simulation purposes we will use pressure of the oil phase P_o and saturation of the water component S_w as the primary variables. This is done so that we allow the "water break" scenarios when the capillary pressure is zero. After the substitution of P_w and S_o as functions of P_o and S_w, we get:

[3] often empirically derived

$$\frac{\partial}{\partial x}\left[k\frac{\lambda_o}{B_o}\left(\frac{\partial P_o}{\partial x}-\gamma_o\frac{\partial D}{\partial x}\right)\right]+$$
$$\frac{\partial}{\partial z}\left[k\frac{\lambda_o}{B_o}\left(\frac{\partial P_o}{\partial z}-\gamma_o\frac{\partial D}{\partial z}\right)\right] = \frac{\partial}{\partial t}\left(\phi\frac{1-S_w}{B_o}\right)+\tilde{q}_o, \qquad (5)$$

and

$$\frac{\partial}{\partial x}\left[k\frac{\lambda_w}{B_w}\left(\frac{\partial(P_o-P_c(S_w))}{\partial x}-\gamma_w\frac{\partial D}{\partial x}\right)\right]+$$
$$\frac{\partial}{\partial z}\left[k\frac{\lambda_w}{B_w}\left(\frac{\partial(P_o-P_c(S_w))}{\partial z}-\gamma_w\frac{\partial D}{\partial z}\right)\right] = \frac{\partial}{\partial t}\left(\phi\frac{S_w}{B_w}\right)+\tilde{q}_w, \qquad (6)$$

– a closed system of two PDE for two unknown functions $P_o(x,z,t)$ and $S_w(x,z,t)$ that we will solve in our simulation.

Following is a summary of our assumptions with regard to the *known* functions that appear in (5) and (6):

- $k = k(x,y,z)$: geometric permeability is a known function of the reservoir points; note that in `OOMP_RS` we implement an arbitrary *anisotropic* geometric permeability, while the abstract multiphase framework allows for anisotropic permeability *tensor* with principal axes *unaligned* with the grid axes – see the type-bound procedure `geometry%perm()`;
- $\mu_o = \mu_o(P_o)$: oil viscosity is either constant or a function of oil pressure[4];
- $k_{ro} = k_{ro}(P_o)$: oil relative permeability is either constant or a function of oil pressure;
- μ_w = const: water viscosity is either constant or a function of water pressure;
- k_{rw} = const: water relative permeability is either constant or a function of water pressure;;
- $\phi = \phi(P_o)$: rock porosity is a function of *oil* pressure; we will assume that $\phi = \phi_0 + c_R(P_o - P^{ref})$ where ϕ_0 is the porosity at some *reference* oil pressure P^{ref} and c_R is the *rock compressibility*.
- $P_c = P_c(S_w)$: capillary pressure is given as some tabulated or power function of water saturation;
- $b_o = 1/B_0 = b_o(P_o)$: the inverse of oil formation volume factor is a function of oil pressure; we will assume that $b_o = b_o^{ref} + c_{fo}(P_o - P^{ref})$ where b_o^{ref} is the inverse formation volume factor at some reference pressure P^{ref} and c_{fo} is the fluid compressibility of oil;
- $b_w = 1/B_0 = b_w(P_w)$: the inverse of water formation volume factor is a function of water pressure; we will assume that $b_w = b_w^{ref} + c_{fw}(P_w - P^{ref})$ where b_w^{ref} is the inverse formation volume factor at some reference pressure P^{ref} and c_{fw} is the fluid compressibility of water; in some simulations we may assume $b_w^{ref} = 1,\ c_{fw} = 1$;

[4] viscosity obviously depends on the temperature but thermal effects are neglected in this study

- ρ_o : density of the oil component at some known fixed conditions (i.e., temperature and pressure); this may be the density at the *stock tank conditions* or this may be the density at some reference pressure P^{ref}; if this reference pressure is the same as in the above formula for the oil formation volume factor, then $b_o^{ref} = 1$;
- ρ_o : density of the water component at some known fixed conditions (i.e., temperature and pressure); this may be the density at at some reference pressure P^{ref}; if this reference pressure is the same as in the above formula for the water formation volume factor, then $b_w^{ref} = 1$.

Note that equations (1) and (2)express *mass balance* relation where we get rid of the common factor – the density of the respective component at reference conditions. However, we still require values of the densities at reference conditions for the *gravitational terms*.

We will assume *zero-flow* boundary conditions at the reservoir boundaries[5] – i.e.

$$\begin{aligned}
&\nabla(P_o - \gamma_o D)(x = x_{\max}, z) = \nabla(P_o - \gamma_o D)(x = x_{\min}, z) = \\
&\nabla(P_o - \gamma_o D)(x, z = z_{\max}) = \nabla(P_o - \gamma_o D)(x, z = z_{\min}) = 0, \\
&\nabla(P_w - \gamma_w D)(x = x_{\max}, z) = \nabla(P_w - \gamma_w D)(x = x_{\min}, z) = \\
&\nabla(P_w - \gamma_w D)(x, z = z_{\max}) = \nabla(P_w - \gamma_w D)(x, z = z_{\min}) = 0. \qquad (7)
\end{aligned}$$

Well terms (source or sink) are described later in the report. And finally, we assume the oil pressure and water saturation known at some initial time $t = t_0$:

$$P_o(x, z, t = t_0) = P_o^0(x, z),\ S_w(x, z, t = t_0) = S_w^0(x, z). \qquad (8)$$

Equations (5) and (6) together with the boundary and initial conditions (7,8) pose a *boundary-value problem*, and solving this problem will be the basis of our reservoir simulation.

3 Units

We use *reservoir* or *customary* units as the principal system of measurement units in our simulation. The unit of length in this system is foot (`ft`), mass – pound (`lb=lbm`), force – pound-force (`lbf`), pressure – pound per square inch (`psi`). After we multiply the left- and right-hand sides of equations (5) and (6) by the unit volume for *finite-volume discretization* in the next section, we will apply the following conversion factors to various components in (5) and (6):

- the gravitational constant will be computed as $\frac{1}{12^2}\rho_l b_l \frac{g}{32.2}$ where ρ_l is the density at reference pressure and b_l is the inverse formation volume factor for oil ($l = o$) or water ($l = w$); for the reservoir units, $g = 32.2$;

[5] for a *rectangular* reservoir; note that the present technique can be used without significant modifications for more complex reservoir shapes by making the geometric permeability very small outside of the true reservoir boundaries, however that may adversely impact the numerical convergence

- the *transmissibility/flow rate* conversion factor $\alpha = 0.001127 \approx 1./887.5$ will be applied at the geometric permeability coefficient k in both equations;
- the *accumulation terms* in the right-hand side of (5) and (6) will be divided by the conversion factor of 5.615 to account for the conversion from the standard cubic feet to standard barrels as all production and injection rates in our simulation are specified in barrels.

We do not need to explicitly incorporate the *constant* conversion factors into our equations so long as we apply them during the implementation. Note, however, that the gravitational constant depends on pressure of the corresponding phase, and hence requires special treatment during the formation of the *Jacobian matrix*. We could get rid of the gravitational terms by introducing new variables, however, that would result in additional complications when manipulating reservoir parameters that are functions of pressure[6].

4 Discretization

In version 2 of `OOMP_RS` reservoir simulator we use the simplified assumption of a *block-centered* spatial grid (see [2])[7]. The abstract framework `multiphase.F90` allows for arbitrary non-uniform *rectangular block-centered* grids (type `geometry`), while the specific implementation `twophase.F90` provides a uniform-grid implementation with $\Delta x \neq \Delta y \neq \delta z$ in the type `regular`. Note, however, that even the specific two-phase implementation of the simulation routines do not directly use the type `regular` and hence we can easily depart from the uniformity constraint by replacing `regular` with an alternative descendant of `geometry`. Note the same holds true for the other key simulator classes: `oilwater` (extends `fluids`), `psat` (extends `model`), `singlecompletion` (extends `wells`) and `twophase_jacobian` (extends `jacobian`). Each *implementation* is directly accessed only by self and the master program, otherwise any specific functionality not implemented in `multiphase.F90` is invoked via *polymorphism* (see [8]).

It was one of the requirements of Phase 3a that a detailed description of the discretization for equations (5,6) be provided, specifying the computation of *Jacobian* components and the corresponding *residual* both for the *Fully Implicit* (FIM) and *Implicit Pressure-Explicit Saturation* (IMPES) methods. As will be evident from subsequent sections, the approach implemented in our reservoir simulator is different from direct discretization of (5,6):

> We recast the reservoir simulation problem as a more general *multiphase flow simulation* with *arbitrary state variables*, and discretize the resulting *redundant* equations. A discretization of (5,6) is then achieved by *substituting* the actual state variables (oil pressure and water saturation), flow and well terms into the more general system.

[6] that would become more complicated functions of the new variables
[7] as opposed to arbitrary *unstructured* grids

Albeit the general N-component multiphase counterpart of system (5,6) is more challenging to solve than the two-phase system, the discretization of the multiphase system is *easier to implement numerically* as it requires less context-dependent branching. However, at the numerical solution stage, the general multiphase Jacobian and residual are transparently reduced to their much smaller two-phase counterparts and thus we take advantage of the simplifications offered by the two-phase model.

Despite the fact that our discretization approach is different from direct application to (5,6), we still describe two discretization approaches for the reduced system – mostly to demonstrate the difference with our approach and reveal the sparsity pattern of the resulting Jacobian.

Following [2], we discretize (5) using *first-order differentiation* in time and *second order differentiation*[8] in space as follows:

$$\begin{aligned}
&(\Upsilon_o^x)_{i-.5,j}\left[(P_o^{n+1})_{i-1,j}-(P_o^{n+1})_{i,j}-(\gamma_o)_{i-.5,j}(D_{i-1,j}-D_{i,j})\right]+\\
&(\Upsilon_o^x)_{i+.5,j}\left[(P_o^{n+1})_{i+1,j}-(P_o^{n+1})_{i,j}-(\gamma_o)_{i+.5,j}(D_{i+1,j}-D_{i,j})\right]+\\
&(\Upsilon_o^z)_{i,j-.5}\left[(P_o^{n+1})_{i,j-1}-(P_o^{n+1})_{i,j}-(\gamma_o)_{i,j-.5}(D_{i,j-1}-D_{i,j})\right]+\\
&(\Upsilon_o^z)_{i,j+.5}\left[(P_o^{n+1})_{i,j+1}-(P_o^{n+1})_{i,j}-(\gamma_o)_{i,j+.5}(D_{i,j+1}-D_{i,j})\right]=\\
&\frac{\Delta x\Delta y\Delta z}{\Delta t}\left[-(\phi b_0)^{n+1}(S_w^{n+1}-S_w^n)+(1-S_w^n)(b_o^{n+1}c_R+\phi^n c_{fo})(P_o^{n+1}-P_o^n)\right],
\end{aligned} \tag{9}$$

where the oil *transmissibilities* in x and z directions are defined as follows:

$$\begin{aligned}
(\Upsilon_o^x)_{i-.5,j} &= \left(\frac{k\Delta y\Delta z}{\Delta x}\times\frac{k_{ro}}{B_o(P_o^k)\mu_o(P_o^k)}\right)_{x=x_i-\delta x_i^-,z=z_j},\\
(\Upsilon_o^x)_{i+.5,j} &= \left(\frac{k\Delta y\Delta z}{\Delta x}\times\frac{k_{ro}}{B_o(P_o^k)\mu_o(P_o^k)}\right)_{x=x_i+\delta x_i^+,z=z_j},
\end{aligned} \tag{10}$$

and

$$\begin{aligned}
(\Upsilon_o^z)_{i,j-.5} &= \left(\frac{k\Delta y\Delta x}{\Delta z}\times\frac{k_{ro}}{B_o(P_o^k)\mu_o(P_o^k)}\right)_{x=x_i,z=z_j-\delta z_j^-},\\
(\Upsilon_o^z)_{i,j+.5} &= \left(\frac{k\Delta y\Delta x}{\Delta z}\times\frac{k_{ro}}{B_o(P_o^k)\mu_o(P_o^k)}\right)_{x=x_i,z=z_j+\delta z_j^+},
\end{aligned} \tag{11}$$

where e.g. δx_i^- and δx_i^+ are the distances from the x-coordinate of the centre of the i-th block x_i to the left and right block boundaries, respectively. For a block-centered grid, $\delta x_{i-1}^+ + \delta x_i^- = x_i - x_{i-1}$[9] at internal block boundaries. The meaning of the pressure time step k is explained below. All the terms in the right-hand side of (9) that are evaluated at time step $n+1$ are computed using the pressure at time step $n+1$. It is important that k may not be equal to $n+1$, and this will later lead to a discussion of various flavors of the IMPES method.

Note that the unknown oil pressure P_o^{n+1} may appear on both sides of (9), both in linear and *nonlinear* terms. Hence, accurate solution for the pressure would require the application of some nonlinear equation solver.

[8] second order for uniform grids and constant coefficients only

[9] with similar equalities along other axes

Likewise for the discretized water conservation equation we have:

$$
\begin{aligned}
&(\Upsilon_w^x)_{i-.5,j}\left[(P_o^{n+1})_{i-1,j}-(P_o^{n+1})_{i,j}-(\gamma_w)_{i-.5,j}(D_{i-1,j}-D_{i,j})\right]+\\
&(\Upsilon_w^x)_{i+.5,j}\left[(P_o^{n+1})_{i+1,j}-(P_o^{n+1})_{i,j}-(\gamma_w)_{i+.5,j}(D_{i+1,j}-D_{i,j})\right]+\\
&(\Upsilon_w^z)_{i,j-.5}\left[(P_o^{n+1})_{i,j-1}-(P_o^{n+1})_{i,j}-(\gamma_w)_{i,j-.5}(D_{i,j-1}-D_{i,j})\right]+\\
&(\Upsilon_w^z)_{i,j+.5}\left[(P_o^{n+1})_{i,j+1}-(P_o^{n+1})_{i,j}-(\gamma_w)_{i,j+.5}(D_{i,j+1}-D_{i,j})\right]-\\
&(\Upsilon_w^x P_c'(S_w^m))_{i-.5,j}\left[(S_w^m)_{i-1,j}-(S_w^m)_{i,j}\right]-\\
&(\Upsilon_w^x P_c'(S_w^m))_{i+.5,j}\left[(S_w^m)_{i+1,j}-(S_w^m)_{i,j}\right]-\\
&(\Upsilon_w^z P_c'(S_w^m))_{i,j-.5}\left[(S_w^m)_{i,j-1}-(S_w^m)_{i,j}\right]-\\
&(\Upsilon_w^z P_c'(S_w^m))_{i,j+.5}\left[(S_w^m)_{i,j+1}-(S_w^m)_{i,j}\right]=\\
&\frac{\Delta x\Delta y\Delta z}{\Delta t}\left\{\left[(\phi b_w)^{n+1}-S_w^n\phi^n c_{fw}P_c'(S_w^n)\right](S_w^{n+1}-S_w^n)+\left[S_w^n b_w^{n+1}c_R+S_w^n\phi^n c_{fw}\right](P_o^{n+1}-P_o^n)\right\},
\end{aligned}
\tag{12}
$$

with water transmissibilities defined as

$$
\begin{aligned}
(\Upsilon_w^x)_{i-.5,j} &= \left(\frac{k\Delta y\Delta z}{\Delta x}\times\frac{k_{rw}}{B_w(P_c(S_w^m)-P_o^k)\mu_w}\right)_{x=x_i-\delta x_i^-,z=z_j},\\
(\Upsilon_w^x)_{i+.5,j} &= \left(\frac{k\Delta y\Delta z}{\Delta x}\times\frac{k_{rw}}{B_w(P_c(S_w^m)-P_o^k)\mu_w}\right)_{x=x_i+\delta x_i^+,z=z_j},
\end{aligned}
\tag{13}
$$

and

$$
\begin{aligned}
(\Upsilon_w^z)_{i,j-.5} &= \left(\frac{k\Delta y\Delta x}{\Delta z}\times\frac{k_{rw}}{B_w(P_c(S_w^m)-P_o^k)\mu_w}\right)_{x=x_i,z=z_j-\delta z_j^-},\\
(\Upsilon_w^z)_{i,j+.5} &= \left(\frac{k\Delta y\Delta x}{\Delta z}\times\frac{k_{rw}}{B_w(P_c(S_w^m)-P_o^k)\mu_w}\right)_{x=x_i,z=z_j+\delta z_j^+}.
\end{aligned}
\tag{14}
$$

All the terms in the right-hand side of (12) that are evaluated at time step $n+1$ are computed using the pressure at time step $n+1$ and saturation at time step m.

If we choose $m=k=n+1$ in (9-14) then solving the resulting system of non-linear equations is called the *Fully Implicit* method (FIM). If we choose $m=n$ but $k=n+1$, we obtain a *variant* of the *Implicit Pressure Explicit Saturation* method (IMPES) where the transmissibilities are computed *implicitly*. This variant of IMPES has been implemented in our simulator.

Alternatively, if $k=m=n$, we obtain a flavor of IMPES with the transmissibilities in (10,11,13,14) being computed *explicitly*. Note that the *accumulation terms* in the right-hand sides of (9) and (12) are still computed for the pressure at $n+1$. This flavor of IMPES would be easier to implement due to the easier computation of the resulting Jacobian; however, we do implement this simplified algorithm in our simulator.

The above equations manifestly lack source-sink (i.e., well) terms. These will be added to the right-hand sides in the subsequent sections, but it is important to note that the well terms are evaluated using oil pressure at time step $n+1$ and saturation at time step m (i.e., $n+1$ for FIM and n for IMPES).

At this point we have all the discretized finite difference equations to numerically solve the boundary-value problem for the oil-water reservoir system. The boundary conditions (7) are treated trivially by dropping the first term in the square brackets in (5,6) when $i = 1$, and the second term when $i = N_x$. Likewise, we drop the third term in square brackets in (5,6) when $j = 1$, and the fourth term when $j = N_z$. Well production/injection rates $(\tilde{q}_o, \tilde{q}_w)$ are evaluated and added to the right-hand side of equations (9,12) at each time step. Although the proposed discretization is only one of many viable alternatives, we will now focus on implementation details for solving these finite-difference equation. However, this scheme is only first order accurate in time, $O(\Delta t)$, and one obvious improvement is to use *Crank-Nicholson* (CN) scheme for time-differentiation. Although CN is of higher order it does not add computational effort to the implicit solver as no new implicit terms are introduced, while allowing larger time steps due to the higher accuracy. However, we will not pursue this further in this project.

5 Operator Notation

Before describing the solution techniques employed in our simulator, we will cast equations (9) and (12) into an operator form that will simplify the subsequent discussion, provide a few notational shortcuts and help demonstrate the analogy with the approach taken in our multiphase framework. First, we introduce our *model* space as the space of oil pressure and water saturation functions discretized on an $N_x \times N_z$ spatial grid *without specifying a time grid*. Below we denote vectors of the model space via $\mathbf{m}$:

$$\begin{aligned}\mathbf{m} \;=\; [&(P_o)_{1,1}(S_w)_{1,1} \ldots (P_o)_{N_x,1}(S_w)_{N_x,1} \\ &(P_o)_{1,2}(S_w)_{1,2} \ldots (P_o)_{N_x,2}(S_w)_{N_x,2} \\ &\ldots \\ &(P_o)_{1,N_z}(S_w)_{1,N_z} \ldots (P_o)_{N_x,N_z}(S_w)_{N_x,N_z}]\end{aligned} \tag{15}$$

Note we order the model by rows (i.e., the index of x is the fastest changing index). Although $\mathbf{m}$ has the natural structure of a matrix, we effectively consider it as an $M = 2 \times N_x \times N_z$-component vector. Next, we represent (9) and (12) as a *nonlinear operator equation* on the space of model vectors (15). Although the involved operators are nonlinear, we represent them as *matrices* applied on model vectors but will allow the components of these matrices to *depend* on the unknown model:

$$\mathbf{O}(\mathbf{m}^{n+1})\mathbf{m}^{n+1} \;=\; \mathbf{R}(\mathbf{m}^n), \tag{16}$$

where $\mathbf{m}^{n+1}$ is the unknown model (at the next time step) and $\mathbf{m}^n$ is known. The right-hand side of (16) is some computable vector-function of known model. In our problem the *data space* (image of the operator) has the same dimension as the model space, since the number of equations equals the number of unknowns and we will seek discretizations that avoid ill-posedness by ensuring the maximum rank ($2 \times N_x \times Nz$) of the Jacobian

of **O**. For computational purposes we split operator **O** and equation (16) as follows:

$$\mathbf{F}(\mathbf{m}^{n+1})\mathbf{m}^{n+1} - \mathbf{g}(\mathbf{m}^{n+1}) = \mathbf{A}(\mathbf{m}^{n+1})\left(\mathbf{m}^{n+1} - \mathbf{m}^{n}\right) + \mathbf{q}(\mathbf{m}^{n+1}), \quad (17)$$

where $\mathbf{F}, \mathbf{A}$ are $M \times M$ matrices with components dependent on the unknown model $\mathbf{m}^{n+1}$, $\mathbf{g}$ is an M-vector with components also functions of the unknown model $\mathbf{m}^{n+1}$, and $\mathbf{q}$ is an M-vector that represent source terms as functions of the unknown model. Operator **F** in (17) represents the *flow terms* from (9) and (12) – i.e., the terms that involve the transmissibilities but exclude the gravitational terms. Operator **A** represents the *accumulation terms* from the right-hand side of the discretized equations, with the time increments of the oil pressure and water saturations appropriately factored out. Vector **g** represents all the terms from (9) and (12) that include component gravities. Assuming that the operator equations (17) are formed by writing out (12) *after* (9) for each grid point, moving along the rows, the operator **A** becomes a *block-diagonal* matrix with 2×2 blocks of the form:

$$\mathbf{D}_L^L = \mathbf{D}_L^L(P_o^{n+1}) = \frac{\Delta x \Delta y \Delta z}{\Delta t} \times \quad (18)$$
$$\begin{bmatrix} (1 - S_w^n)\left[b_o(P_o^{n+1})c_R + \phi(P_o^n)c_{fo}\right] & -b_o(P_o^{n+1})\phi(P_o^{n+1}) \\ S_w^n\left[\tilde{b}_w(P_o^{n+1})c_R + \phi(P_o^n)c_{fw}\right] & \tilde{b}_w(P_o^{n+1})\phi(P_o^{n+1}) - S_w^n\phi(P_o^n)c_{fw}P_c'(S_w^n) \end{bmatrix}$$

where $L = (j-1)N_x + i = 1, \ldots, M$ and all the quantities are computed at grid point (i, j) and $\tilde{b}_w(P_o^{n+1}) = b_w(P_c(S_w^{n+1}) - P_o^{n+1})$. Note the order of equations in our notation is different from [2], and we already took advantage of the known *linear* dependence of the inverse formation volume factors and porosity on pressure. Operator **F** is block-banded with 5 non-zero block-diagonals[10] $\mathbf{T}_L^L, \mathbf{T}_{L\pm1}^L, \mathbf{T}_{L\pm N_x}^L$ given by

$$\mathbf{T}_L^L = \mathbf{T}_L^L(P_o^l, S_w^m) = \quad (19)$$
$$\begin{bmatrix} -\sum(\Upsilon_o^x)_{i\pm.5,j} - \sum(\Upsilon_o^z)_{i,j\pm.5} & 0 \\ -\sum(\Upsilon_w^x)_{i\pm.5,j} - \sum(\Upsilon_w^z)_{i,j\pm.5} & \sum(\Upsilon_w^x P_c')_{i\pm.5,j} + \sum(\Upsilon_w^z P_c')_{i,j\pm.5} \end{bmatrix}$$

$$\mathbf{T}_{L\pm1}^L = \mathbf{T}_{L\pm1}^L(P_o^l, S_w^m) = \quad (20)$$
$$\begin{bmatrix} (\Upsilon_o^x)_{i\pm.5,j} & 0 \\ (\Upsilon_w^x)_{i\pm.5,j} & -(\Upsilon_w^x P_c')_{i\pm.5,j} \end{bmatrix}$$

and

$$\mathbf{T}_{L\pm N_x}^L = \mathbf{T}_{L\pm N_x}^L(P_o^l, S_w^m) = \quad (21)$$
$$\begin{bmatrix} (\Upsilon_o^z)_{i,j\pm.5} & 0 \\ (\Upsilon_w^z)_{i,j\pm.5} & -(\Upsilon_w^z P_c')_{i,j\pm.5} \end{bmatrix}$$

We linearize (17) around $\mathbf{m}^{n+1}$ for Newton iterations. This effectively means computing the *Jacobian* of all the terms in (17) that depend on

[10] we avoid calling such matrices "pentadiagonal" as the latter term is usually associated with matrices that have bandwidth 5; in our cases **F** is a sparse matrix with 5 non-zero block diagonals up to $2N_x$ positions apart

$\mathbf{m}^{n+1}$. IN case of *single-completion* wells, the Jacobian $\mathbf{J}$ always has the block structure of operator $\mathbf{F}$ as equations (9) and (12) indicate that the pressure and saturation at any given block depends on pressure and saturation *only in its neighbors and self*. Consequently, neighbors along the x axis contribute to the 2×2 blocks $\mathbf{J}^L_{L\pm 1}$, and neighbors along the z axis contribute to blocks $\mathbf{J}^L_{L\pm N_x}$ of the Jacobian. The accumulation term only contributes to the diagonal blocks $\mathbf{J}^L_L$. The gravitational terms contribute to the off-diagonal blocks $\mathbf{J}^L_{L\pm 1}$ of Jacobian, because gravities at block boundaries are computed as arithmetic means of the gravities in the adjacent blocks ([2]). Partial derivatives of the transmissibilities may contribute to the off-diagonal Jacobian blocks if *the flow is from the adjacent blocks* – i.e., based on the *upwinding* condition ([1],[2]).

6 Multiphase Formulation

Our multiphase generalization of the *discretized* two-phase model (9-14), is based on the following assumptions:

1. A reservoir state is uniquely determined by a model vector $\mathbf{m} = (\xi^1, \xi^2, \ldots, \xi^N)$ (N – the number of phases in the system) defined at every point of a 3D grid; if the grid has $N_x \times N_y \times N_z$ points (block centers) then the reservoir state is determined by $N \times N_x \times N_y \times N_z$ variables;
2. In a *discretized*[11] form, the material balance equation for the phase number l between the n-th and $n+1$-st time steps is given by

$$\sum_{k=1}^{N} \sum_{B \in \mathrm{NB}(J)} R^{lk}_{upw(B,J,F^k_{BJ})} \Upsilon^{geom}_{BJ} \Upsilon^{k}_{upw(B,J,F^k_{BJ})} F^k_{BJ} \;=\; \sum_{k=1}^{N} a^{lk}_J \frac{\xi^{n+1,k}_J - \xi^{n,k}_J}{\Delta t} + q^l_J(\xi),$$

$$F^k_{BJ} \;=\; P^k_B - P^k_J - 0.5(\gamma^k_B + \gamma^k_J)(D_B - D_J), \qquad (22)$$

   where $J = (i_x, i_y, i_z)$ is a 3-index of a reservoir block ranging through all blocks, $B \in \mathrm{NB}(J)$ are the 3-indices of the adjacent blocks

$$(i_x \pm 1, i_y, i_z), (i_x, i_y \pm 1, i_z), (i_x, i_y, i_z \pm 1);$$

3. In (22) P^k_I is the pressure of phase k in block I, γ^k_I is the gravity of phase k in block I, R^{lk}_I is the solubility of phase l in phase k computed in grid block I, Υ^k_I is the fluid transmissibility of phase k computed in block I, and $\Upsilon^{\mathrm{geom}}_{BJ}$ is the *geometric* transmissibility computed at the interface between blocks J and B, which does not depend on the state variables;
4. The well term $q^l_J(\xi)$ represents a phase-k source/sink in block J, and for general multi-completion wells has a non-local dependence on the state variables for all phases at time step $n+1$ $\xi^{n+1,\cdots}_{\cdots}$; however, for single-completion wells this term will be shown to depend on $\xi^{n+1,1}, \ldots, \xi^{n+1,N}$ locally in block J only;

[11] note that we *formulate* the multiphase model in a discrete form, rather than obtaining it from PDE's by discretization

5. The *upwind* indexing function is defined as follows:

$$upw(B, J, \eta) \;=\; \begin{cases} B & \text{if } \eta \geq 0, \\ J & \text{otherwise} \end{cases} \tag{23}$$

   (note that this function's value is a 3-index – i.e. a 3D vector with integer components);

6. The *accumulation coefficients* a_J^{lk} depend on the state variables in block J only;

7. Any functions in (22) that depend on the state variables ξ are evaluated at time step $n+1$ (fully implicit formulation);

8. No-flow boundary conditions are imposed along the reservoir boundaries; the *potential differences* F_I^k in (22) are assigned zero values when the block index vector I has any component exceeding the allowed range.

Equations (22) for $l = 1, \ldots, N$ and J ranging across all grid blocks form a system of $N \times N_x \times N_y \times N_z$ equations for as many state variables $\xi_J^{n+1,l}$ at the next time step. In our simulator, this system is solved at each time step by Newton iterations (see e.g. [9]) by linearizing the nonlinear equations around an *approximation* to the next step model ξ^{n+1}, starting with $\xi_{\text{INIT}}^{n+1} = \xi^n$ as the initial approximation.

7 Jacobian Computation and Packing

Since our objective is to develop a simulator for *immiscible* oil and water phases and for notational convenience, we will assume in this section that the components are not soluble in each other – i.e., $R^{lk} = \delta^{lk}$ in (22) and the first sum in the left-hand side of (22) can be dropped. Note that this is not a critically important simplification for our multiphase framework module `multiphase.F90`, and extending to fully miscible systems with an arbitrary number of components would require only minor changes to the multiphase module.

In the Newton method, we iteratively compute approximations to the next-step model $\mathbf{m}^{n+1} \approx \mathbf{m}^{n+1,i},\ i = 1, 2, \ldots$ by solving linear systems

$$\mathbf{J}(\mathbf{m}^{n+1,i})(\mathbf{m}^{n+1,i+1} - \mathbf{m}^{n+1,i}) \;=\; \mathbf{R}(\mathbf{m}^{n+1,i}), \tag{24}$$

starting with $\mathbf{m}^{n+1,0}$ and stopping when the model update is less than the prescribed accuracy. The residual vector in the right-hand side of (24) is computed by substituting $\xi^{n+1} = \mathbf{m}^{n+1,i}$ into (22) and pulling all the terms to the right-hand side. The Jacobian is the Jacobian matrix of the nonlinear operator obtained by pulling all the terms in (22) to the left-hand side and computing *all* the partial derivatives of the *all* the $N \times N_x \times N_y \times N_z$ "coordinate components" of the nonlinear operator with respect to as many model (state) variables, after substituting $\xi^{n+1} = \mathbf{m}^{n+1,i}$. This potentially huge matrix $\mathbf{J}$ is, however, extremely sparse, with $7 \times N$ non-zero diagonals, *not counting the contributions from the well terms.* This number comes from contributions due to flows between each *internal*

block and 6 adjacent blocks (in 3D) that have a common boundary with the given block; since, in the general case of multiphase flows, the fluid transmissibilities and phase pressures for each phase *may depend on all state variables* – as when multi-component capillary pressures depend on multiple saturations (see e.g. [4]) – contributions from each of the 7 blocks may result in up to N non-zero Jacobian diagonals. If the state variables are grouped in a model vector as in (15) component-first packing order used in Section 5, then the Jacobian effectively becomes a 7-block-diagonal matrix with $N \times N$ blocks.

If (24) is solved using some *exact sparse linear solver* that takes advantage of the *block-sparsity* of $\mathbf{J}$ (see e.g. [6]), then the above block structure proves useful. Note, however, that for realistic grid sizes $N_{xyz} \approx 1000$ the exact solver approach might be unacceptable as the total cost of solving (24) once is const $\times N^2 \times N_x \times N_y \times N_z$ with a (large) constant independent of N and the grid sizes (see e.g. [6]). Another, and potentially even greater, disadvantage of using exact solvers is that efficient exact solvers require *component-wise access* to $\mathbf{J}$, thus making their application heavily dependent on the actual memory packing of the Jacobian.

In our Phase 1 report and version 1 of the simulator, we have used a banded solver (`gbtrf/gbtrs`) from the `LAPACK` library ([3]) applied directly to solving the 2D Jacobian equations, or in a *splitting* arrangement solving a series of 1D problems (see e.g. [2]). The latter approach, although it can effectively assuage the computational complexity of solving large-scale systems, unfortunately introduces other issues, such as *loss of accuracy* (see [1]) and, depending on the splitting mechanism employed, *non-conservative* behaviour (see [7]).

In version 2 of the simulator we have completely departed from the approach based on using exact solvers, and have implemented a *Krylov-subspace iterative solver* (see [10],[6]) for solving (24). The advantages of using an iterative solver can be summed up in the order decreasing of importance as follows:

1. the initial approximation to $\mathbf{m}^{n+1}$ is always "reasonably close" to the true solution in the sense that a Krylov solver would take considerably fewer steps[12] to converge than when starting from an *arbitrary* initial approximation (maximum of $N \times N_x \times N_y \times N_z$, see [10]); an exact solver would perform the same amount of arithmetic operations regardless of the initial approximation;
2. our iterative solver (type `ls_cgnr` in `cgnr.F90`, extends abstract `ls_solver` in `base_solver.F90`) does not require component-wise access to $\mathbf{J}$ and can use a black-box application of Jacobian operator (and its adjoint) through the type-bound procedure `ls_oper%apply()` in `base_oper.F90`;
3. because the Jacobian application is a "black box" operation for the solver, we have completely encapsulated the multiphase reservoir model in the abstract type `model` in `multiphase.F90` derived from the abstract type `ls_vector` in `base_vector.F90`, thus decoupling the back-end solver from the front-end state-vector implementation;

[12] here as "steps" we refer to the iterations of the linear solver, not the Newton iterations

4. the CGNR solver (see [6]) implemented in our solver module `cgnr.F90` can be transparently replaced by any other iterative solver implementation (e.g., GMRES, see [6],[10]) so long as that implementation is derived from the abstract type `ls_solver` in `base_solver.F90` and takes abstract `ls_oper` and `ls_vector` class arguments.

The actual computation of the residual is performed in the *abstract* type `fluids` in `multiphase.F90` by the type-bound procedure `fluids%residual` and in the *abstract* type `jacobian` in `multiphase.F90` by the type-bound procedure `jacobian%update`. Note that the latter is the *only* Jaconbian type-bound procedure implemented in `multiphase.F90`, all the other procedures are deferred till the actual implementation (in the type `twophase_jacobian.F90` in `twophase.F90`. This is critically important for our approach as we effectively delegate the task of computing the Jacobian to the abstract parent of the two-phase class, thus effectively taking advantage of the conceptual simplicity of the multiphase model (22). The type-bound procedure `jacobian%update()` accesses abstract type-bound procedures of abstract types `model, geometry,` and `wells` that are dynamically replaced with the actual two-phase implementations at run time. If we would like to implement an alternative model – e.g., a black-oil simulator – we would simply need to supply *black-oil implementations* for the various components in (22) in classes derived from `jacobian`, `fluids` and `model`, without having to update any other parts of the framework.

Jacobian computation in `jacobian%update` is performed in a row-by-row loop[13] by *pulling* non-zero columns into each row (see e.g. [5]). If $upw(B, J, F^k_{BJ}) \geq 0$ then the transmissibility derivative (times the corresponding potential difference) is contributed to off-diagonal Jacobian terms with the diagonal determined by multi-index B. When (15) packing and "natural" subsurface coordinates are used[14], the terms are contributed to a diagonal[15] $N_x \times N_y$ positions above/below the main diagonal if B points to lower/upper blocks, N_x positions above/below the main diagonal if B points to southern/northern grid blocks, and 1 position above/below the main diagonal if B point to eastern/western blocks. Each of the flow terms F^k_{BJ} contributes partial derivatives to both diagonals associated with indices B and J as described above, and the accumulation terms contribute partial derivatives to the main (block) diagonal only.

Once all the non-zero columns for each row of the Jacobian are identified in `jacobian%update`, a deferred type-bound procedure `jacobian%pd` is invoked to insert the evaluated partial derivative (argument `v`) at the identified row (arguments `i,ix,iy,iz`) and column (arguments `j,ixd,iyd,izd`). Note that since `jacobian%pd` is deferred, the actual updating is handled by a descendant of the abstract type, and the multiphase framework need not be aware of the specifics of memory allocation and ordering employed in the implementation.

Note that the same residual and Jacobian computation procedures are used both for FIM and IMPES. In the latter case, the Jacobian in-

[13] "row" here means each state variable index

[14] z points down, x east, and y south

[15] actually, a block diagonal; however, here we will drop the reference to "block" to avoid confusion with grid blocks

vokes a deferred type-bound function `jacobian%explicit(k)` that indicates whether the component (phase) number k state variable is computed explicitly (value 1) or implicitly. If the former, the partial derivatives with respect to that component for all of the left-hand side terms in (22) are assumed zero; the partial derivatives with respect to that component of the accumulation terms a_J^{lk} are assumed zero as well. However, the linear terms $\xi_J^{n+1,k} - \xi_J^{n,k}$ are differentiated with respect to $\xi_J^{n+1,k}$ regardless of the value returned by `jacobian%explicit(k)`. This is done to effectively allow *automatic elimination* of the explicit components from (24) in the linear solver[16]. The residual computation routine `fluids%residual` takes an "IMPES" and "substitution" flags. If the IMPES flag is on, it computes the residual with the "explicit" components taken from the previous time step everywhere except in the linear terms $\xi_J^{n+1,k} - \xi_J^{n,k}$. If the substitution flag is set, the linear terms use the updated state variables for the explicitly computed components, otherwise these terms use the values from the previous time step as well. The reason for this dual-flag approach is that *we do not segregate the residual for the pressure-only part of (24)* because of the *black-box nature of our linear solver*. In terms of our two-phase IMPES implementation, even though at initial and intermediate Newton iterations the water saturation "updates" generated by solving the IMPES version of (24) are useless and should not be applied to the saturations from the previous step until *the pressure iterations have converged*, we use these intermediate saturation updates to compute the residual of (24) to check for the convergence of the oil pressure. However, once the pressure iterations have converged, we apply the saturation update to the water saturation from the previous time step, as (24) at the final Newton iteration combines in itself a quasilinearized parabolic or elliptic pressure equation and a hyperbolic saturation equation.

The following code snippet demonstrates the main time evolution loop for both FIM and IMPES. The code is an excerpts from `p3_tests.F90` driver program that was specifically developed for Phase 3b reference/test case demonstrations.

```
! initially m_next = m_interm = m_prev = model (state) at time step t
  do
!
!   save the saturations from the previous
!   time step if using IMPES
    if (impes==1) then
      prevsat=m_interm.get(2)
    end if
!
!   Newton-Raphson iterations
    inewton=0
    do
!     update the flow (directions) based on m_interm
      call fl.update(m_interm,f,grid)
!
```

[16] note that we do not eliminate the saturation explicitly – or triagonalize – the Jacobian but rely on the linear solver to do this automatically

```
!     compute the Jacobian...
      call jac.update(m_next, m_prev,m_interm,fl,f,grid,w,t,dt,impes)
!
!     and the residual; if IMPES, substitute the updated saturations
!     in the linear part of the RHS to compute the residual norm
      call f.residual(r,m_next,m_prev,m_interm,grid,w,fl,t,dt,impes,1)
!
      if (r.linfnorm()/m_prev.linfnorm()< tol .or. inewton==nnewton) then
!        call displaymodel(r,'NR convergence residual')
        exit
      else
        inewton=inewton+1
      end if
!
!     if IMPES, recompute the residual using the saturations from the
!     previous time step
      if (impes==1) &
        call f.residual(r,m_next,m_prev,m_interm,grid,w,fl,t,dt,impes,0)
!
!     initialize the iterative Krylov solver
      allocate(solver)
      call solver.init(jac, m_interm, r,.false.)
!
!     run Krylov solver iterations for ONE STEP of Newton-Raphson
!     until the Krylov solver stalls (in ~NSIZE steps)
      do
        call solver.step()
        if (solver.stalled()) exit
!        dm=solver.m
      end do
!
!     get the model update
      dm=solver.m
!
      deallocate(solver)
!
!     update the model iterate

      call m_next.lc(m_interm,dm,ONE,ONE)
      m_interm=m_next
      if (impes==1) then
        call m_interm.set(2,prevsat)
      end if
!
    end do ! Newton iterations
!
!
    t=t+dt
    if (t>maxtime) exit
  end do
```

Note that we allow the Krylov solver to run until it *stalls* – the latter happens when the *Arnoldi/Lanczos iterations* have reached an eigenspace of the linear operator (see [10],[6]). In our context such a situation may mean

- convergence to the true solution (observed in all of the tests carried out so far);
- the Jacobian has a very high condition number, effectively resulting in the numerical rank lower than the true rank.

The latter situation may arise in case of extremely stiff discretizations [1], and is especially relevant for general multiphase models. The simplest remedy would be to carry on with the residual computation, and because the Newton iterations fail to converge, reduce the time step and hence the stiffness of the system. This approach is implemented in `p3_tests.F90`. More generic approaches to improving the convergence of Krylov solvers for ill-conditioned problems (see [10],[6]) include e.g. *reorthogonalization* (implemented in our solver module `cgnr.F90`) and pre-conditioning. Further discussion of this subject goes beyond the scope of this project.

8 Well Model

Our treatment of production/injection wells is based on the following:

- the multiphase framework implemented in our module `multiphase.F90` allows for *multiple-completion* vertical or horizontal wells in the abstract type `wells`;
- the two-phase implementation in our module `twophase.F90` provides *single-completion* vertical-only production/injection wells;
- pressures of all the phases inside a well are equal, based on the natural observation that the oil and water phase pressures at water break must be equal (because the capillary pressure for zero saturation of the wetting phase is zero – [4]);
- both the abstract framework and specific implementation allow well control parameters based on
  1. a specified time-dependent production/injection rate for the well in barrels at *stock tank* conditions for any one component; the control component may change with time (e.g., oil production followed by water injection);
  2. a specified bottom-hole pressure at completion number 1; if this is specified together with the flow rate, the flow rate overrides this parameter;
  3. maximum and/or minimum bottom-hole pressure at the completion number one for each well; this constraint can be combined with the flow rate constraint;
- the multiphase framework allows completion perforation anywhere within a grid block; the two-phase implementation assumes that the completion is at the center of the block;

- arbitrary rectangular blocks and anisotropic geometric permeability[17] are allowed;
- different well skin factors can be provided for different completions of the same well;
- deviated wells are neither provided for in the abstract type `wells`, nor implemented in `singlecompletion`.

With the above assumptions/constraints in mind, we proceed to the formulation of the well model.

The steady-state 2D pressure distribution for a radially symmetric flow of a fluid with a viscosity μ, injection rate q^{well} from a cylinder of radius r^{well} (well completion) in a medium of height Δz with permeability k, can be described by the solution to the Laplace equation in the polar coordinates and is given by

$$p(r) \;=\; p^{well} + \frac{q^{well}\mu}{2\pi k \Delta z} \ln \frac{r}{r_{well}}, \tag{25}$$

where p^{well} in (25) is the fluid pressure at the cylinder wall. Now assuming that we are given some "average" pressure in a block, p_o, the "equivalent radius" r_o is the radius at which the analytic solution (25) equals p_o. Key to the integration of the analytic solution (25) with numerical schemes is *expressing the equivalent radius via known reservoir geometric parameters*, such as bock dimensions, and reducing (25) for $r = r_o$ to

$$q^{well} \;=\; \frac{2\pi k \Delta z}{\mu} \frac{1}{\ln \frac{r_o}{r_{well}}} (p_o - p^{well}). \tag{26}$$

For a multiphase flow, taking into account phase mobilities and assuming all phase pressures equal inside the well, we can express (26) as

$$\begin{aligned} q_l^{well} \;&=\; \mathrm{WI} \times \frac{k_{rl} b_l}{\mu_l} (p_l - p^{well}), \\ \mathrm{WI} \;&=\; \frac{2\pi k \Delta z}{\mu} \frac{1}{\ln \frac{r_o}{r_{well}} + s}, \end{aligned} \tag{27}$$

where l is the phase/component index, P_l is the pressure of phase l in the grid block containing the well completion, WI is referred to as *well index* and s is the well *skin factor*. Note that the skin factor may be regarded as a normalizing or corrective term and does not arise from the simple analytic derivation given above. Multiplication of the phase l mobility by the inverse formation volume factor b_l is due to the assumption that all flow rates are specified at the stock-tank conditions. From (27) we can relate the flow rates at the completion for different phases:

$$q_i^{well} \;=\; \frac{k_{ri} b_i \mu_l}{\mu_i k_{rl} b_l} q_l^{well}. \tag{28}$$

Note that (28) holds even before the "break out" of phase i: if the saturation of phase i is zero, then the physically meaningful relative permeability function k_{ri} is zero as well, indicating zero flow rate for phase i component.

[17] arbitrary diagonal permeability tensor

In order to incorporate equations (27) and (28) into our simulator, we only require an expression for the equivalent radius via the grid and reservoir geometry. We use the following semi-heuristic approximation due to Peaceaman ([2]):

$$r_o \approx 0.28 \frac{\sqrt{\sqrt{k_1}\Delta x^2 + \sqrt[-1/2]{k_1}\Delta y^2}}{\sqrt[1/4]{k_1} + \sqrt[-1/4]{k_1}},\ k_1 = \frac{k_y}{k_x}, \tag{29}$$

where k_x, k_y are the rock permeabilities along x, y axes, respectively. We implement (29) in `oilwater%q, oilwater%dq` type-bound procedures of the type `oilwater` in `twophase.F90` derived from the abstract type `fluids`.

In case of multiple completions, the flow rates in (27) will have to be summed up to produce the total well flow rate, and the well pressures at different completions will be related by a hydrostatic gradient and, if friction effects are incorporated, loss of pressure due to friction. Consequently, unless completions are in the adjacent blocks, the well constraint equations relate state variables in distant blocks. As a result, the Jacobian may have additional non-zero diagonals and lose the symmetric 7-diagonal structure. Multiple-completion source terms are accounted for in the Jacobian computation routine `jacobian%update()` in `multiphase.F90`. For each *additional* completion, the multiphase module allows adding up to N extra diagonals to the Jacobian. However, the `twophase.F90` implementation module provides only for a single completion per well, and the well terms in this implementation contribute only to the diagonal elements of the Jacobian.

We will now describe the detailed application of well controls at each time step in the type-bound procedures `oilwater%q` and `oilwater%dq`, and how that affects the residual and Jacobian computation in `jacobian%update`.

- the deferred type-bound procedure `wells%controls()` returns the well controls for each well at a specified time t;
- at each Newton iteration, well rates for each component are computed inside the grid block that contains a well completion;
- the "well control component" phase (i.e. oil for oil producing well, etc.) for which a flow rate is specified, is added to the right hand side of (22) and the Jacobian rows corresponding to the control component and the containing block are not updated (unless a bottom-hole condition is violated – see below); the well pressure is computed from (27);
- if the upper or lower pressure bound is violated, the well pressure is set to that bound;
- if the bottom-hole pressure was specified or set due to the violation of pressure bounds, the flow rate for the control component is (re)computed using (27);
- if flow rate is (re)computed, then the Jacobian is updated with the partial derivatives of the right-hand side of (27) for all phases and the containing block;

- if the flow is not computed from the well pressure, then only the rows of the Jacobian corresponding to the non-control component and the containing grid block are updated with the partial derivatives of (28).

9 Phase 3b Analysis

In this section we present phase 3b case studies. In addition to providing the results of the initial Jacobian and residual computation, we carry out a stability and convergence analysis of both FIM and IMPES.

9.1 Reference Case With Gravity

Run `p3_test.x`, specify test 1, specify 0 maximum time, 0.5 initial time step, 0.5 minimum time step, arbitrary NR tolerance and 1 as maximum Newton-Raphson iterations, select FIM as the solution method (0). The resulting Jacobian will be output in an executable `Matlab` script file `case1.m`. The most significant elements of the Jacobian are -7126661 and 7480782, and agree well with the provided reference case (-7128100 and 7482500) accounting for the lower accuracy of the provided `Matlab` result and the ordering scheme used in our simulator. Other elements are $< 1.e+6$ and are provided at too low accuracy for exact comparison but are the right order of magnitude with respect to our results. *The Residual produced by our code and the one provided for this exercise do not match.*

The next subsection *validates our residual and Jacobian computation* without regard to the provided reference values.

9.2 Reference Case With Gravity – run to convergence

Run `p3_test.x`, specify test 1, specify 365 as the maximum time, 1 initial time step, 1 minimum time step, 0.0001 NR tolerance and 10 as maximum Newton-Raphson iterations, select FIM as the solution method (0).

The resulting oil pressures are

4099.970	4099.970	4099.970
4264.617	4264.617	4264.617
4430.108	4430.108	4430.108

and water saturations

0.094	0.094	0.094
0.100	0.100	0.100
0.106	0.106	0.106

Our driver program outputs the pressure difference between the centers of blocks (1,1) and (1,2), and (1,2) and (1,3):

```
hydrostatic pressure due to 550ft of emulsion:   164.646628861526        165.491316421459
```

The above figures agree with the hydrostatic pressure differential due to 550 `ft` column of 90% oil and 10% water emulsion:

$$550 \times (40 \times 0.9 + 62 \times 0.1)/144 \approx 160\texttt{psi}.$$

Note that the initial pressure difference was ≈ 60 `psi` hence our solution is converging to a physically meaningful configuration. Also note how the water saturation is decreasing in the top layer and increasing at the bottom, while conserving the total amount of water.

Running the same test with a smaller time step produces the same result.

10 years run time:

```
SPECIFY CASE TO RUN (1,2,3):1
MAX TIME (=0 FOR PH3B):3650
INITIAL (and maximum) TIME STEP:1
MINIMUM TIME STEP:1
NR tolerance:0.001
MAX NEWTON-RAPHSON ITERATIONS (=1 FOR PH3B):10
IMPES/FIM (=1/0):0
.............................................
Final model oil comp:4106.669  4106.669  4106.669  4268.952  4268.952  4268.952
   4433.534  4433.534  4433.534
 20130313 040953.438:
 Final model water comp:0.060     0.060     0.060     0.084     0.084     0.084
     0.156     0.156     0.156
 20130313 040953.439:
 hydrostatic pressure due to 550ft of emulsion:   162.283232351721        164.58
 1745602047
```

– and again, a very good agreement with the underlying physical model, with water continuously migrating to lower levels. Note, however, that due to capillary effects the downward migration of water, while continuing, slows down.

Stability test with a very large time step (10 years):

```
SPECIFY CASE TO RUN (1,2,3):1
MAX TIME (=0 FOR PH3B):3650
INITIAL (and maximum) TIME STEP:3650
MINIMUM TIME STEP:3650
NR tolerance:0.001
MAX NEWTON-RAPHSON ITERATIONS (=1 FOR PH3B):1000
IMPES/FIM (=1/0):0
.............................................
 Final model oil comp:4096.816  4096.816  4096.816  4258.516  4258.516  4258.516
   4422.223  4422.223  4422.223
 20130313 041324.892:
 Final model water comp:0.052     0.052     0.052     0.070     0.070     0.070
     0.178     0.178     0.178
 20130313 041324.893:
 hydrostatic pressure due to 550ft of emulsion:   161.700350796471        163.70
 6514736158
```

The result for the large time step cannot be expected to be accurate, but it demonstrates the stability of our FIM solver. The pressures are still very accurate due to the diffusive effect's of the large time step agreeing with the underlying physical process.

100 years run time:

```
SPECIFY CASE TO RUN (1,2,3):1
MAX TIME (=0 FOR PH3B):36500
INITIAL (and maximum) TIME STEP:1
MINIMUM TIME STEP:1
NR tolerance:0.001
MAX NEWTON-RAPHSON ITERATIONS (=1 FOR PH3B):100
IMPES/FIM (=1/0):0
.............................................
 Final residual oil comp:-1.705    -1.705    -1.705    -2.735    -2.735    -2.73
 5     4.439     4.439     4.439
 20130313 041957.258:
 Final residual water comp:1.622     1.622     1.622     2.600     2.600     2.6
 00    -4.221    -4.221    -4.221
 20130313 041957.259: ========================================================
 20130313 041957.259:
 Final model oil comp:4117.095  4117.095  4117.095  4277.838  4277.838  4277.838
   4439.243  4439.243  4439.243
 20130313 041957.259:
 Final model water comp:0.016     0.016     0.016     0.025     0.025     0.025
     0.259     0.259     0.259
```

1000 years run time:

```
SPECIFY CASE TO RUN (1,2,3):1
MAX TIME (=0 FOR PH3B):3650000
INITIAL (and maximum) TIME STEP:10
MINIMUM TIME STEP:0.1
NR tolerance:0.001
MAX NEWTON-RAPHSON ITERATIONS (=1 FOR PH3B):100
IMPES/FIM (=1/0):0
.............................................
 Final residual oil comp:-1.705    -1.705    -1.705    -2.734    -2.734    -2.73
 4     4.439     4.439     4.439
 20130313 042828.210:
 Final residual water comp:1.622     1.622     1.622     2.599     2.599     2.5
 99    -4.221    -4.221    -4.221
 20130313 042828.210: ========================================================
 20130313 042828.210:
 Final model oil comp:4117.238  4117.238  4117.238  4277.981  4277.981  4277.981
   4439.386  4439.386  4439.386
 20130313 042828.210:
 Final model water comp:0.016     0.016     0.016     0.025     0.025     0.025
     0.259     0.259     0.259
 20130313 042828.210:
 hydrostatic pressure due to 550ft of emulsion:   160.742747986578        161.40
 5162435065
```

– we can clearly see water "locked" in the upper layers at some low saturation levels due to capillary effects.

IMPES test:

```
SPECIFY CASE TO RUN (1,2,3):1
```

```
MAX TIME (=0 FOR PH3B):365
INITIAL (and maximum) TIME STEP:1
MINIMUM TIME STEP:0.1
NR tolerance:0.001
MAX NEWTON-RAPHSON ITERATIONS (=1 FOR PH3B):10
IMPES/FIM (=1/0):1
.............................................
 Final residual oil comp:0.002     0.002     0.002     0.000     0.000     0.000
     -0.002    -0.002    -0.002
 20130313 043206.331:
 Final residual water comp:0.000     0.000     0.000     0.000     0.000     0.0
 00     0.000     0.000     0.000
 20130313 043206.331: ============================================================
 20130313 043206.331:
 Final model oil comp:4099.674  4099.674  4099.674  4264.831  4264.831  4264.831
   4430.289  4430.289  4430.289
 20130313 043206.331:
 Final model water comp:0.093     0.093     0.093     0.100     0.100     0.100
     0.106     0.106     0.106
 20130313 043206.331:
 hydrostatic pressure due to 550ft of emulsion:   165.157246548627        165.45
 7342620851
```

– a very good agreement with our FIM result for the same maximum run time.

9.3 Reference Case Without Gravity

Run `p3_test.x`, specify test 2, specify 0 maximum time, 0.5 initial time step, 0.5 minimum time step, arbitrary NR tolerance and 1 as maximum Newton-Raphson iterations, select FIM as the solution method (0). The resulting Jacobian will be output in an executable `Matlab` script file `case2.m`. The most significant elements of the computed and reference Jacobians again agree. *The Residual produced by our code and the one provided for this exercise do not match.*

The next subsection *validates our residual and Jacobian computation* without regard to the provided reference values.

9.4 Reference Case – run to convergence

Run `p3_test.x`, specify test 2, specify 365 as the maximum time, 1 initial time step, 1 minimum time step, 0.0001 NR tolerance and 10 as maximum Newton-Raphson iterations, select FIM as the solution method (0).

```
SPECIFY CASE TO RUN (1,2,3):2
MAX TIME (=0 FOR PH3B):365
INITIAL (and maximum) TIME STEP:1
MINIMUM TIME STEP:1
NR tolerance:0.0001
MAX NEWTON-RAPHSON ITERATIONS (=1 FOR PH3B):10
IMPES/FIM (=1/0):0
```

```
.............................................
 Final residual oil comp:-0.366    -0.366    -0.366     0.000     0.000     0.00
 0     0.367     0.367     0.367
 20130313 043928.064:
 Final residual water comp:-0.022    -0.022    -0.022     0.000     0.000     0.
 000     0.022     0.022     0.022
 20130313 043928.064: ============================================================
 20130313 043928.064:
 Final model oil comp:4264.245  4264.245  4264.245  4264.271  4264.271  4264.271
   4264.296  4264.296  4264.296
 20130313 043928.064:
 Final model water comp:0.100     0.100     0.100     0.100     0.100     0.100
     0.100     0.100     0.100
 20130313 043928.064:
 hydrostatic pressure due to 550ft of emulsion:  2.534296584508411E-002  2.53540
 4676564212E-002
```

– the pressures have equalized throughout the reservoir and the saturations obviously stay constant as expected, in keeping with the physical model (no gravity)

Exactly the same result is produced by IMPES solver (`IMPES=1`). As in case 1, arbitrarily large time steps can be specified, affecting the accuracy of the saturation but not the pressure because of the diffusive nature of pressure equalization.

9.5 Test Case With Gravity

Run `p3_test.x`, specify test 3, specify 0 maximum time, 0.5 initial time step, 0.5 minimum time step, arbitrary NR tolerance and 1 as maximum Newton-Raphson iterations, select FIM as the solution method (0). The resulting Jacobian will be output in an executable `Matlab` script file `case3.m`.

The most significant elements of the computed Jacobian are -7143790, -7146966, 7917944, 7933202. The Residual produced by our code is

```
our_initial_residual(           1 )=  -1388.60632578487      ;
 our_initial_residual(          7 )=   2165.13420245475      ;
 our_initial_residual(         13 )=  -724.836643997927      ;
 our_initial_residual(          3 )=  -1388.60632578487      ;
 our_initial_residual(          9 )=   2165.13420245475      ;
 our_initial_residual(         15 )=  -724.836643997927      ;
 our_initial_residual(          5 )=  -1388.60632578487      ;
 our_initial_residual(         11 )=   2165.13420245475      ;
 our_initial_residual(         17 )=  -724.836643997927      ;
 our_initial_residual(          2 )=   2390.49149921636      ;
 our_initial_residual(          8 )=  -2281.92547044827      ;
 our_initial_residual(         14 )=  -161.854776378351      ;
 our_initial_residual(          4 )=   2390.49149921636      ;
 our_initial_residual(         10 )=  -2281.92547044827      ;
 our_initial_residual(         16 )=  -161.854776378351      ;
 our_initial_residual(          6 )=   2390.49149921636      ;
```

```
 our_initial_residual(          12 )=  -2281.92547044827      ;
 our_initial_residual(          18 )=  -161.854776378351
```

The next subsection *validates our residual and Jacobian computation.*

9.6 Test Case – run to convergence

Run `p3_test.x`, specify test 3, specify 365 as the maximum time, 1 initial time step, 1 minimum time step, 0.0001 NR tolerance and 10 as maximum Newton-Raphson iterations, select FIM as the solution method (0).

```
SPECIFY CASE TO RUN (1,2,3):3
MAX TIME (=0 FOR PH3B):365
INITIAL (and maximum) TIME STEP:1
MINIMUM TIME STEP:1
NR tolerance:0.0001
MAX NEWTON-RAPHSON ITERATIONS (=1 FOR PH3B):10
IMPES/FIM (=1/0):0
.............................................
 Final residual oil comp:0.041     0.041     0.041    -0.018    -0.018    -0.018
     -0.021    -0.021    -0.021
 20130313 045147.926:
 Final residual water comp:-0.003    -0.003    -0.003     0.003     0.003     0.
 003     0.000     0.000     0.000
 20130313 045147.926: ============================================================
 20130313 045147.926:
 Final model oil comp:8846.649  8846.649  8846.649  9056.575  9056.575  9056.575
   9235.240  9235.240  9235.240
 20130313 045147.926:
 Final model water comp:0.445     0.445     0.445     0.146     0.146     0.146
     0.109     0.109     0.109
 20130313 045147.926:
 hydrostatic pressure due to 550ft of emulsion:   209.926583869808        178.66
 4443329662
```

The pressure differences between layers are in a reasonably good agreement with the hydrostatic gradient in a stabilized pressure profile:

$$550 \times (40 \times 0.6 + 62 \times 0.4)/144 \approx 188\texttt{psi},$$
$$550 \times (40 \times 0.6 + 62 \times 0.4)/144 \approx 170\texttt{psi},$$

10 years run time:

```
MAX TIME (=0 FOR PH3B):3650
INITIAL (and maximum) TIME STEP:1
MINIMUM TIME STEP:1
NR tolerance:0.0001
MAX NEWTON-RAPHSON ITERATIONS (=1 FOR PH3B):10
IMPES/FIM (=1/0):0
.............................................
 Final residual oil comp:0.008     0.008     0.008     0.004     0.004     0.004
     -0.012    -0.012    -0.012
```

```
 20130313 045818.136:
 Final residual water comp:-0.001    -0.001    -0.001     0.000     0.000     0.
 000     0.000     0.000     0.000
 20130313 045818.136: ============================================================
 20130313 045818.136:
 Final model oil comp:8894.178  8894.178  8894.178  9079.336  9079.336  9079.336
   9273.294  9273.294  9273.294
 20130313 045818.136:
 Final model water comp:0.181     0.181     0.181     0.227     0.227     0.227
     0.292     0.292     0.292
 20130313 045818.136:
 hydrostatic pressure due to 550ft of emulsion:   185.157261127511        193.95
 8269384751
```

– while the saturation is sufficiently high, water is continuously migrating to lower layers.

100 years run time:

```
SPECIFY CASE TO RUN (1,2,3):36500
 20130313 050112.530: invalid case number - quitting
maharram@glad:src$ ./p3_tests.x
SPECIFY CASE TO RUN (1,2,3):3
MAX TIME (=0 FOR PH3B):36500
INITIAL (and maximum) TIME STEP:1
MINIMUM TIME STEP:1
NR tolerance:0.0001
MAX NEWTON-RAPHSON ITERATIONS (=1 FOR PH3B):100
IMPES/FIM (=1/0):0
.............................................
 Final residual oil comp:0.000     0.000     0.000     0.000     0.000     0.000
      0.000     0.000     0.000
 20130313 050227.599:
 Final residual water comp:0.000     0.000     0.000     0.000     0.000     0.0
 00     0.000     0.000     0.000
 20130313 050227.599: ============================================================
 20130313 050227.599:
 Final model oil comp:8947.536  8947.536  8947.536  9117.883  9117.883  9117.883
   9290.895  9290.895  9290.895
 20130313 050227.599:
 Final model water comp:0.017     0.017     0.017     0.029     0.029     0.029
     0.653     0.653     0.653
 20130313 050227.600:
 hydrostatic pressure due to 550ft of emulsion:   170.346868058248        173.01
 1999770137
```

– and again, water shows signs of slower migration at similar levels of saturation as in case 1. The downward migration will continue but at exponentially slower rates.

100000 years run time:

```
SPECIFY CASE TO RUN (1,2,3):365000
 20130313 050539.117: invalid case number - quitting
```

```
maharram@glad:src$ ./p3_tests.x
SPECIFY CASE TO RUN (1,2,3):3
MAX TIME (=0 FOR PH3B):365000
INITIAL (and maximum) TIME STEP:1
MINIMUM TIME STEP:1
NR tolerance:0.0001
MAX NEWTON-RAPHSON ITERATIONS (=1 FOR PH3B):100
IMPES/FIM (=1/0):0
.............................................
 Final residual oil comp:-0.344    -0.344    -0.344    -0.585    -0.585    -0.58
 5     0.929     0.929     0.929
 20130313 051037.772:
 Final residual water comp:0.310     0.310     0.310     0.526     0.526     0.5
 26    -0.835    -0.835    -0.835
 20130313 051037.772: ============================================================
 20130313 051037.773:
 Final model oil comp:8950.513  8950.513  8950.513  9120.776  9120.776  9120.776
   9291.855  9291.855  9291.855
 20130313 051037.773:
 Final model water comp:0.007     0.007     0.007     0.012     0.012     0.012
     0.681     0.681     0.681
 20130313 051037.773:
 hydrostatic pressure due to 550ft of emulsion:   170.263439009399        171.07
 9093006061
```

– almost all of the water is in the bottom layer, with the water content of oil in the top layer below 1%.

1000 years time-step stability test:

```
SPECIFY CASE TO RUN (1,2,3):3
MAX TIME (=0 FOR PH3B):365000
INITIAL (and maximum) TIME STEP:365000
MINIMUM TIME STEP:365000
NR tolerance:0.0001
MAX NEWTON-RAPHSON ITERATIONS (=1 FOR PH3B):100
IMPES/FIM (=1/0):0
.............................................
 Final residual oil comp:-0.003    -0.003    -0.003    -0.047    -0.047    -0.04
 7     0.048     0.048     0.048
 20130313 052314.515:
 Final residual water comp:0.232     0.232     0.232    -0.049    -0.049    -0.0
 49    -0.184    -0.184    -0.184
 20130313 052314.515: ============================================================
 20130313 052314.515:
 Final model oil comp:8872.134  8872.134  8872.134  9042.230  9042.230  9042.230
   9212.922  9212.922  9212.922
 20130313 052314.515:
 Final model water comp:0.009     0.009     0.009     0.010     0.010     0.010
     0.681     0.681     0.681
 20130313 052314.516:
 hydrostatic pressure due to 550ft of emulsion:   170.096602443478        170.69
```

```
 1988758776
```

– as before, the diffusive effect of the extremely large time step that agreed with the underlying physical process suppressed the errors that could be expected from such a coarse discretization.

IMPES test:

```
SPECIFY CASE TO RUN (1,2,3):3
MAX TIME (=0 FOR PH3B):365
INITIAL (and maximum) TIME STEP:1
MINIMUM TIME STEP:1
NR tolerance:0.0001
MAX NEWTON-RAPHSON ITERATIONS (=1 FOR PH3B):10
IMPES/FIM (=1/0):1
.............................................
 Final residual oil comp:0.129     0.129     0.129    -0.115    -0.115    -0.115
     -0.012    -0.012    -0.012
 20130313 052130.044:
 Final residual water comp:-0.005    -0.005    -0.005     0.005     0.005     0.
 005     0.001     0.001     0.001
 20130313 052130.044: ============================================================
 20130313 052130.044:
 Final model oil comp:8846.866  8846.866  8846.866  9058.892  9058.892  9058.892
   9233.364  9233.364  9233.364
 20130313 052130.044:
 Final model water comp:0.441     0.441     0.441     0.154     0.154     0.154
     0.106     0.106     0.106
 20130313 052130.044:
 hydrostatic pressure due to 550ft of emulsion:   212.026636591881        174.47
 1542935215
```

– the results are comparable to FIM with the same run time, step and accuracy parameters.

10 Phase 4 Test Case 4A

The results of FIM simulation for this test case are shown on Fig 1,2, and the corresponding IMPES results are shown on Fig 3,4. A good qualitative and quantitative agreement can be seen with the results of ECLIPSE simulation on Fig 5,6. The bottom-hole pressure curves produced by the two simulators match for the production well but slightly differ for the injector – see Fig 7. As we will see in later tests, this difference is probably specific to this homogeneous test.

The test case input files for `OOMP_RS` are `tests/4AFIM/4A` and `tests/4AIMPES/4A`; similar naming rules apply to all the tests. Any tests described in this document can be run by executing the `run.sh` Bourne shell script within the corresponding test directory.

11 Phase 4 Test Case 4B

We have performed three types of analysis for this test case:

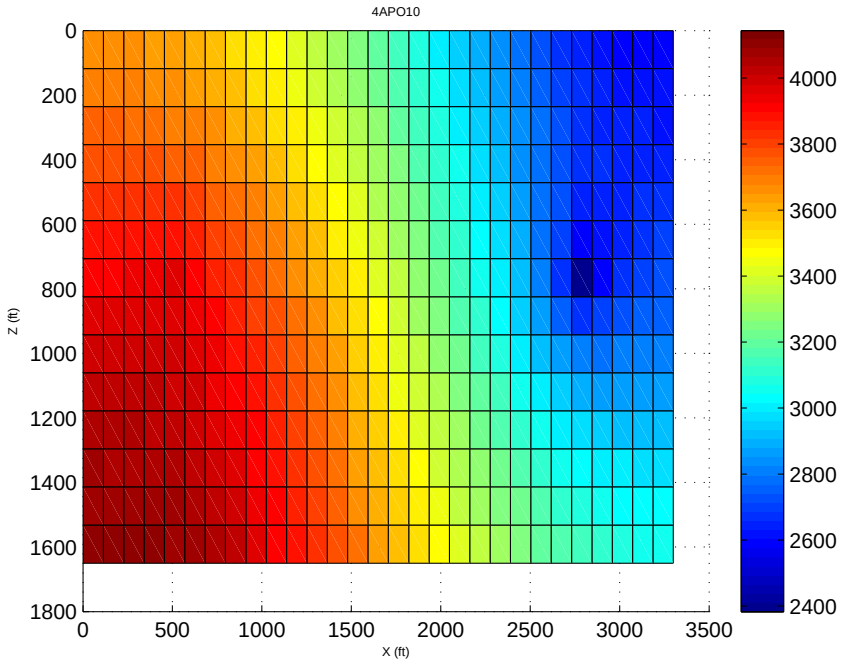

Figure 1: Oil pressure after 5-year FIM simulation – test 4AFIM.

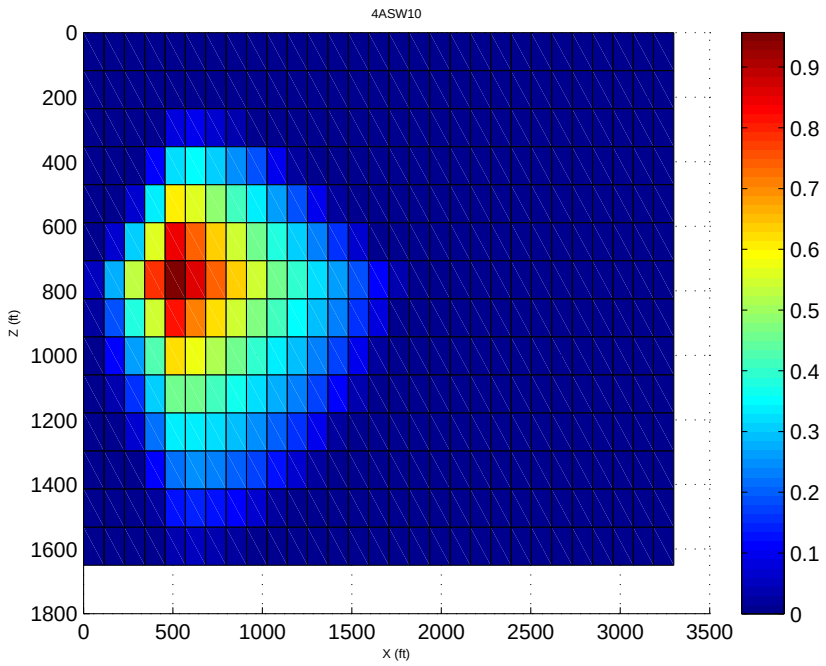

Figure 2: Water saturation after 5-year FIM simulation – test 4AFIM.

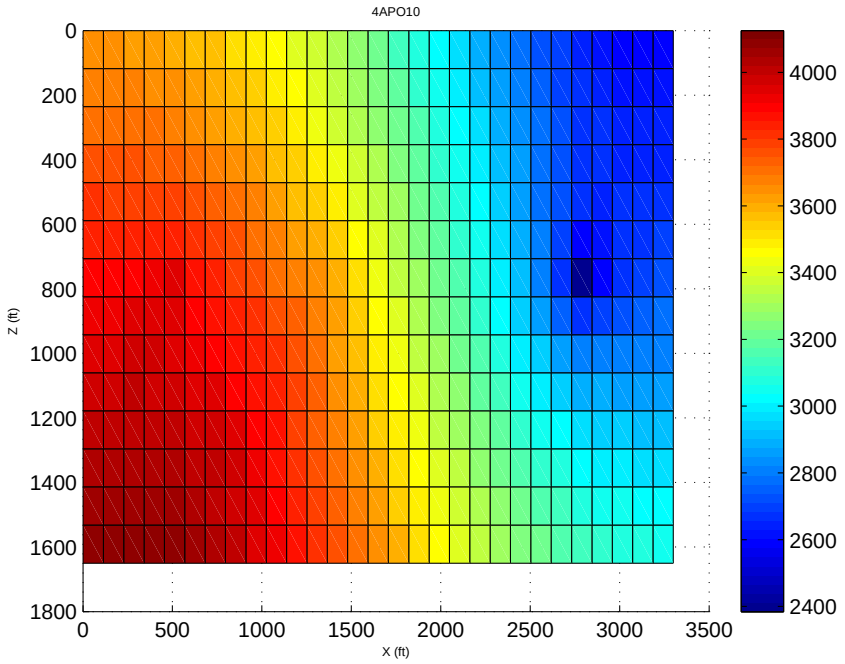

Figure 3: Oil pressure after 5-year IMPES simulation – test 4AIMPES.

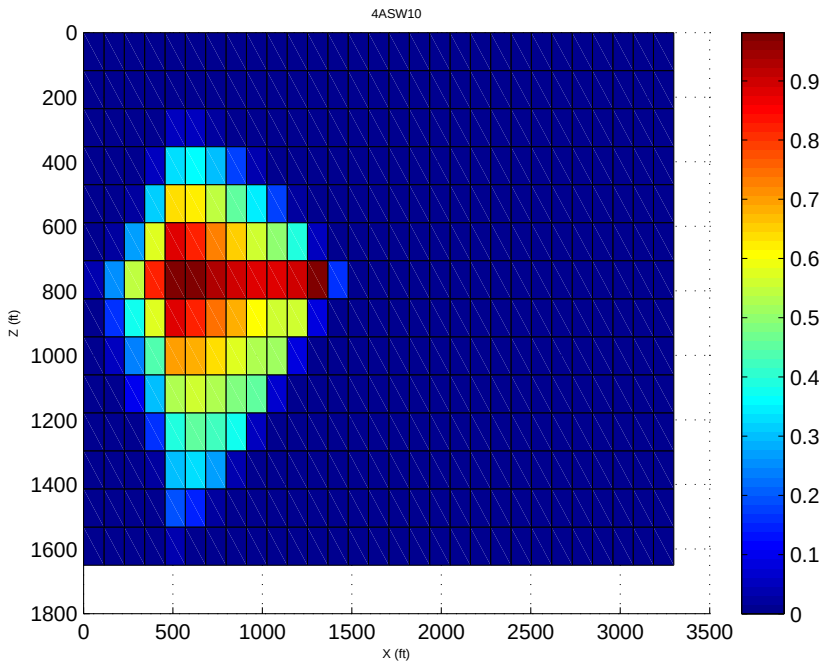

Figure 4: Water saturation after 5-year IMPES simulation – test 4AIMPES.

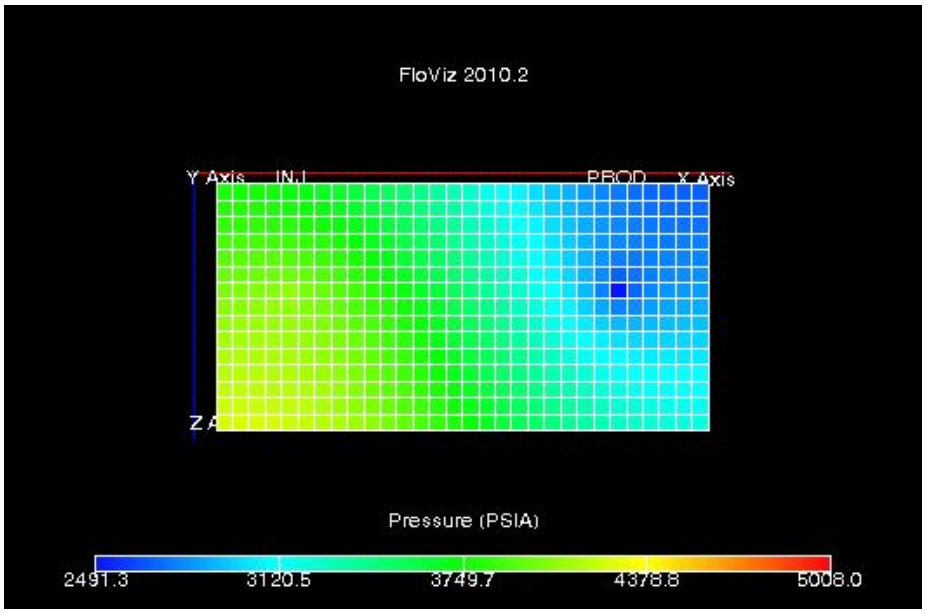

Figure 5: Oil pressure after 5-year ECLIPSE simulation – test 4A.

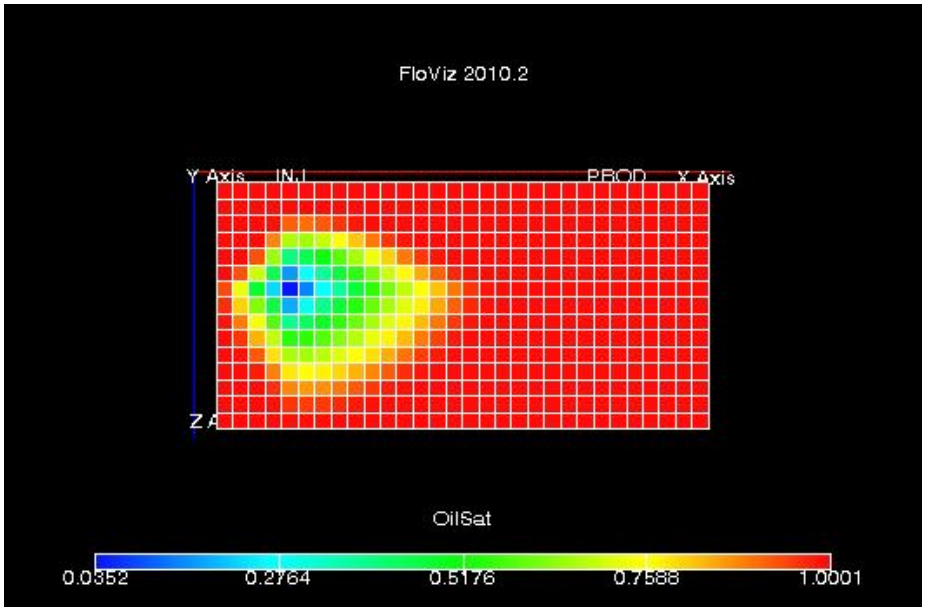

Figure 6: Water saturation after 5-year ECLIPSE simulation – test 4A.

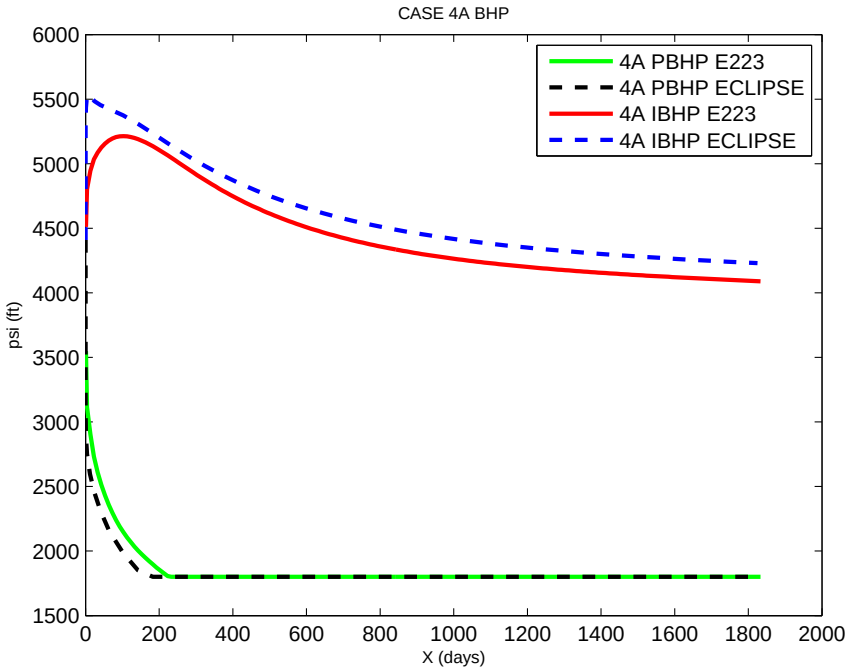

Figure 7: Well BHP after 5-year simulation – test 4AFIM vs ECLIPSE.

- sensitivity to well locations;
- sensitivity to change of (oil) mobility;
- the effect of spatial grid refinement.

Fig 8,9 demonstrated the result of FIM simulation (with 1 day maximum time step) for test case 4B. The corresponding ECLISPE results are shown on Fig 10,11. Note that the bottom-hole pressure curves predicted by the two simulators show good agreement throughout the simulation history (see Fig12), with the `OOMP_RS` showing a more realistic behaviour near zero time, apparently due to the small initial time step used in our simulation (.1 day).

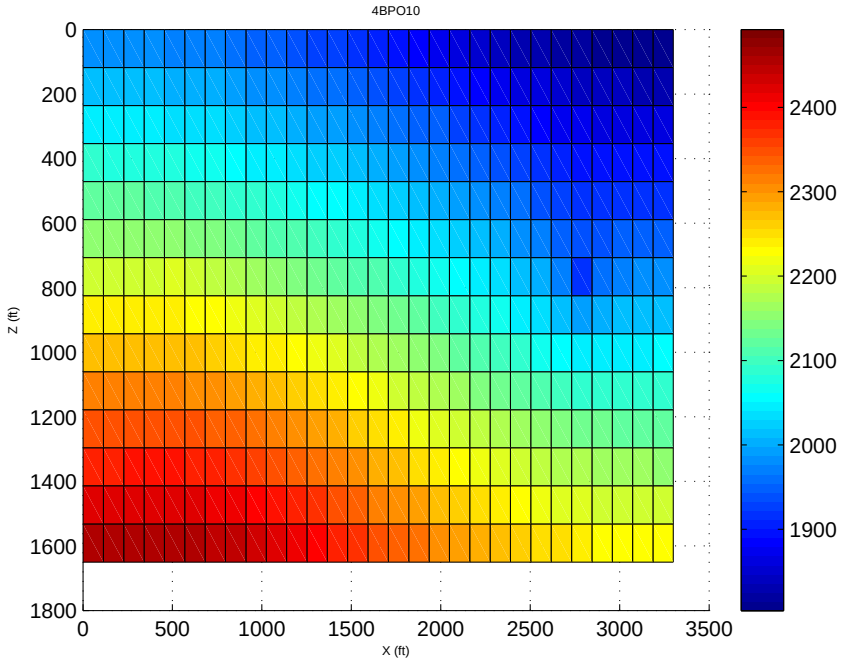

Figure 8: Oil pressure after 5-year FIM simulation – test 4BFIM.

The results of IMPES modeling shown on Fig 15,15 are in good *qualitative*[18] agreement with the results of the FIM modeling, and the predicted

[18] and quantitative for oil pressure

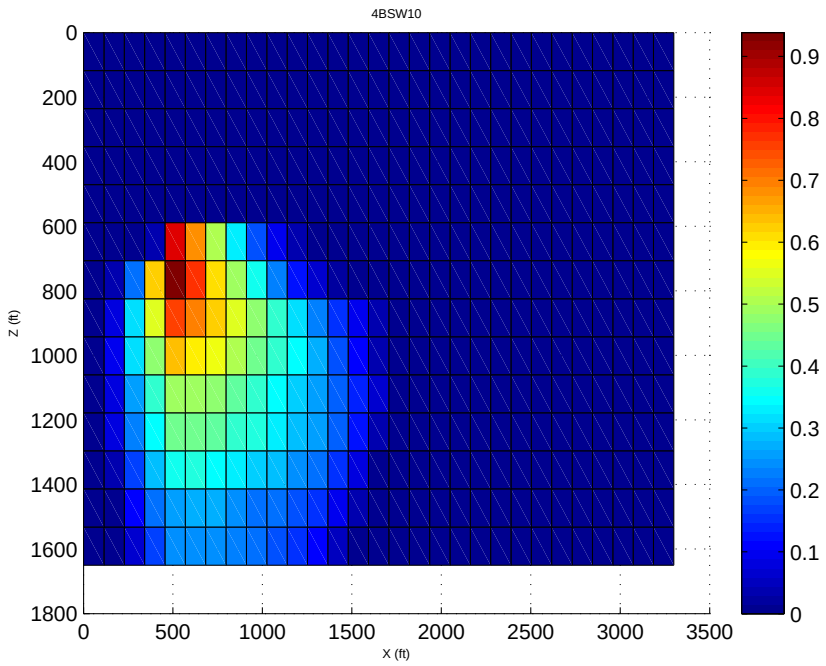

Figure 9: Water saturation after 5-year FIM simulation – test 4BFIM.

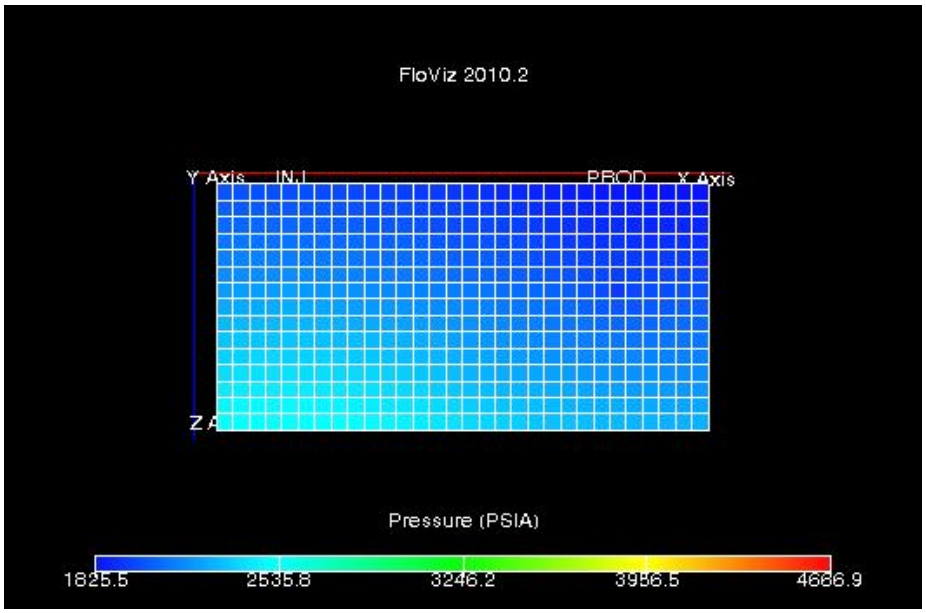

Figure 10: Oil pressure after 5-year ECLIPSE simulation - test 4B.

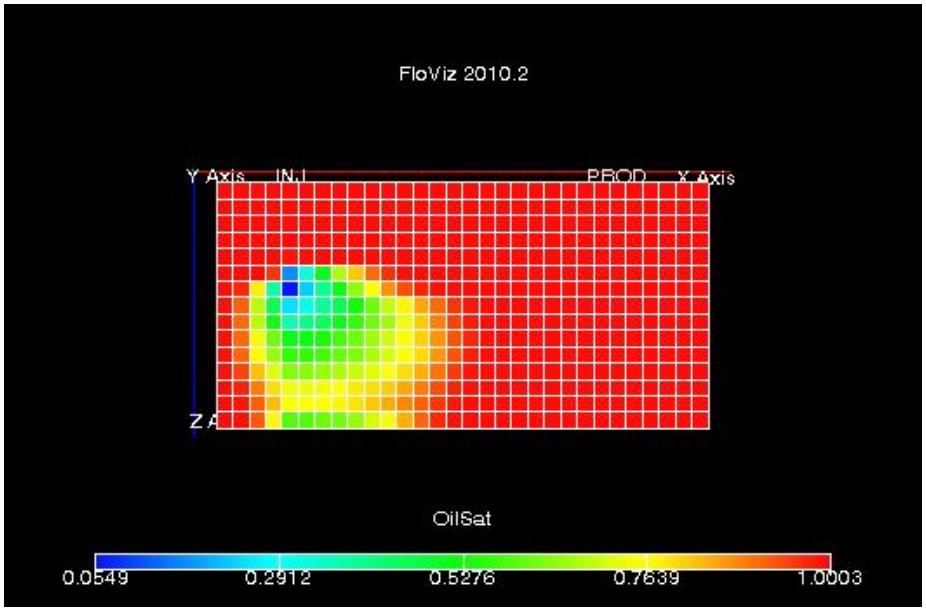

Figure 11: Water saturation after 5-year ECLIPSE simulation – test 4B.

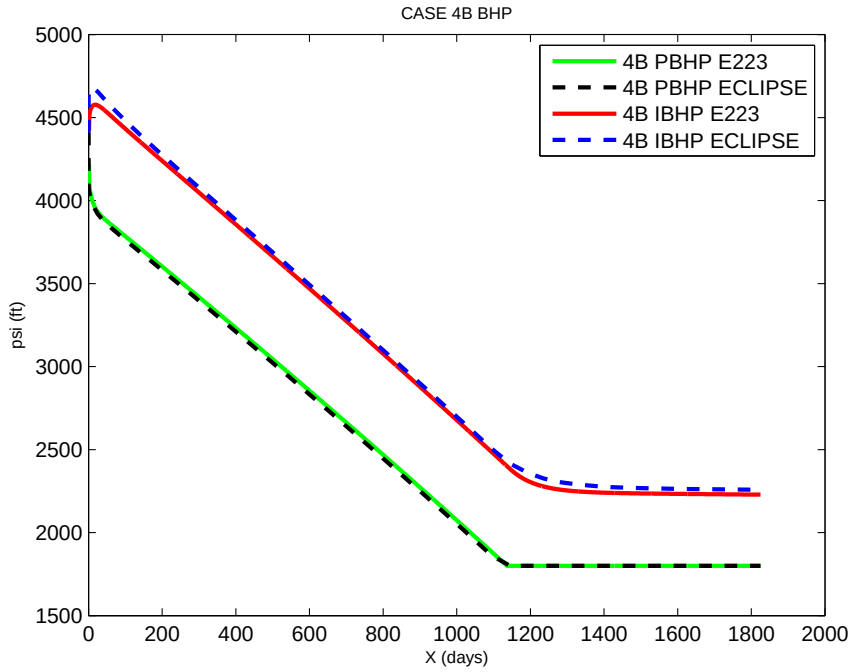

Figure 12: Well BHP – test 4BFIM vs ECLIPSE.

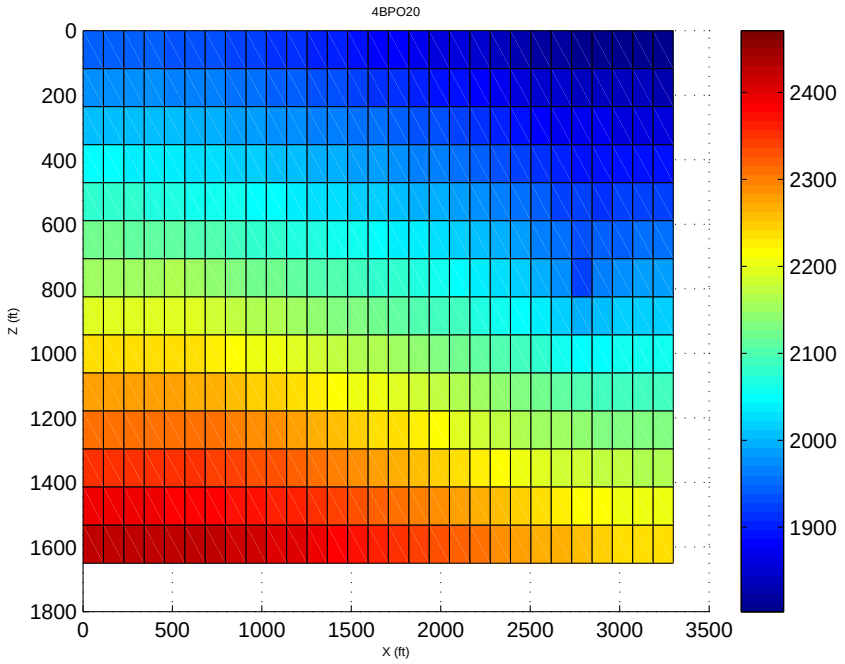

Figure 13: Oil pressure after 10-year FIM simulation – test 4BFIM.

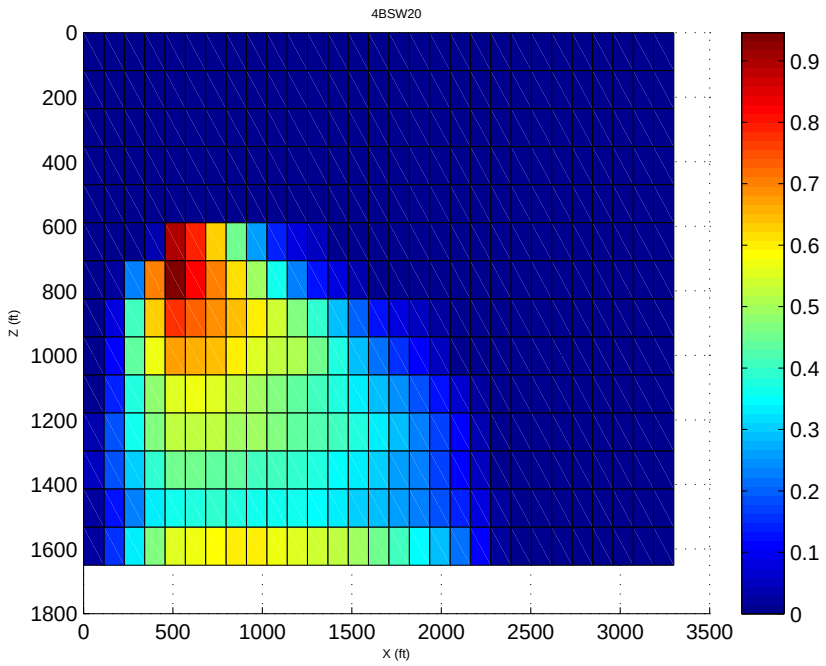

Figure 14: Water saturation after 10-year FIM simulation – test 4BFIM.

BHP curves match (see Fig17). Note how the saturation front modeled by IMPES is *sharper* than in the FIM case, which is in agreement with the saturation "decoupled" from the pressure propagating according to a first-order *hyperbolic* PDE (see [1]). The saturation front takes advantage of the horizontal permeability channel – this effect can be demonstrated using FIM only for higher-resolution grids, as will be shown later in the document.

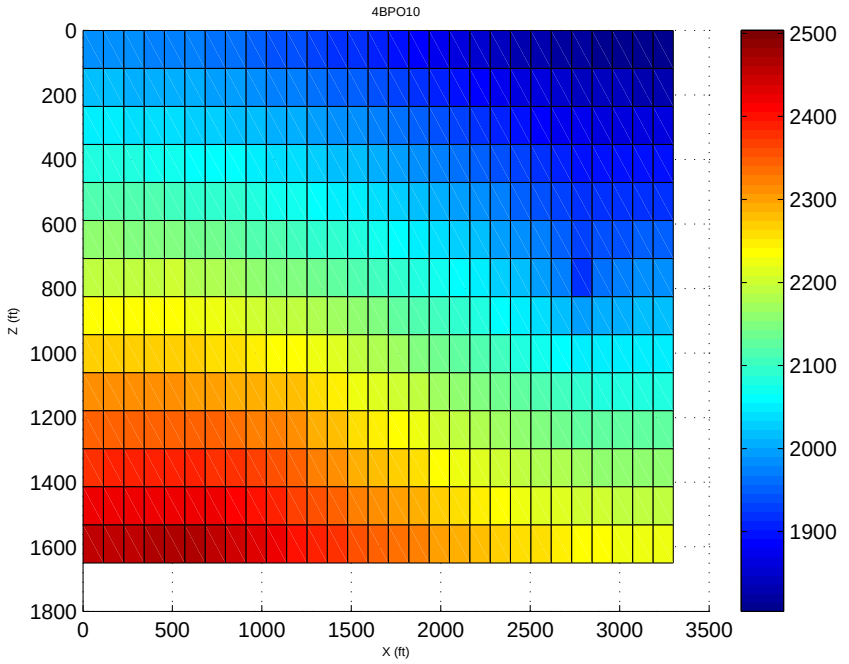

Figure 15: Oil pressure after 5-year IMPES simulation – test 4BIMPES.

Note the qualitative agreement between the *long-term* (10 year) FIM and IMPES modeling by our simulator, shown on Fig 13,14,18,19.

11.1 Well Relocation

In the well location sensitivity analysis, we move the injector upward by 440 `ft`, and pull the producer down 330 ft.The resulting oil pressure and

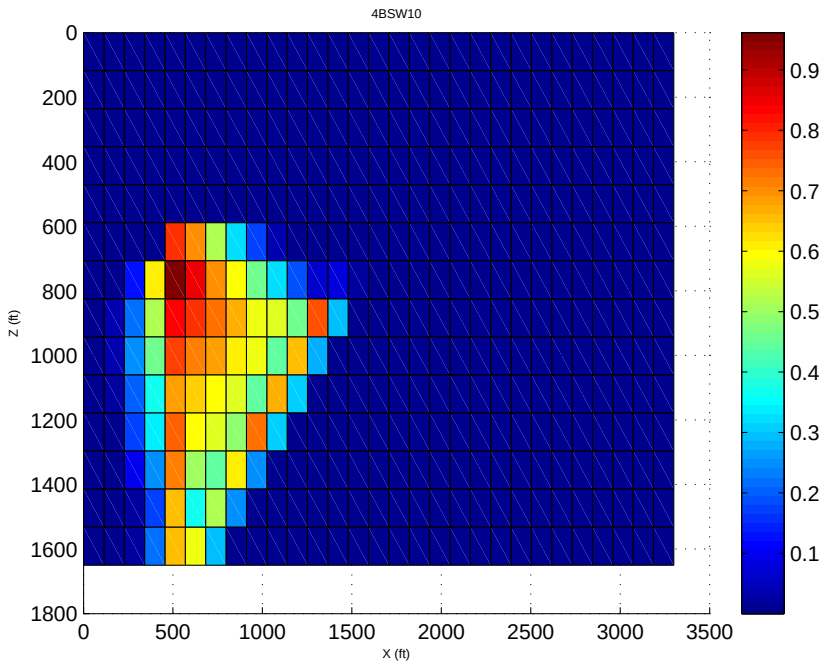

Figure 16: Water saturation after 5-year IMPES simulation – test 4BIMPES.

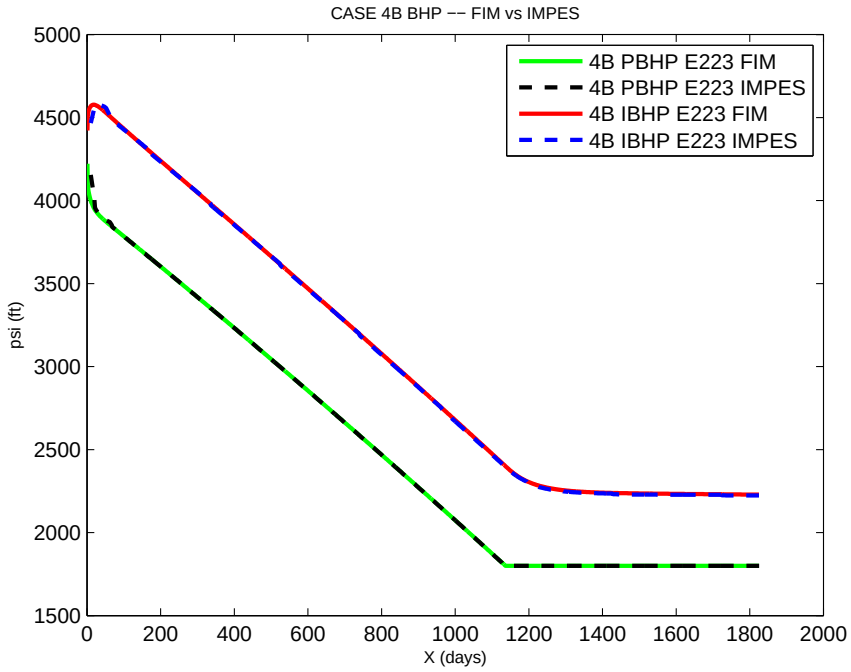

Figure 17: Well BHP – test 4BFIM vs 4BIMPES.

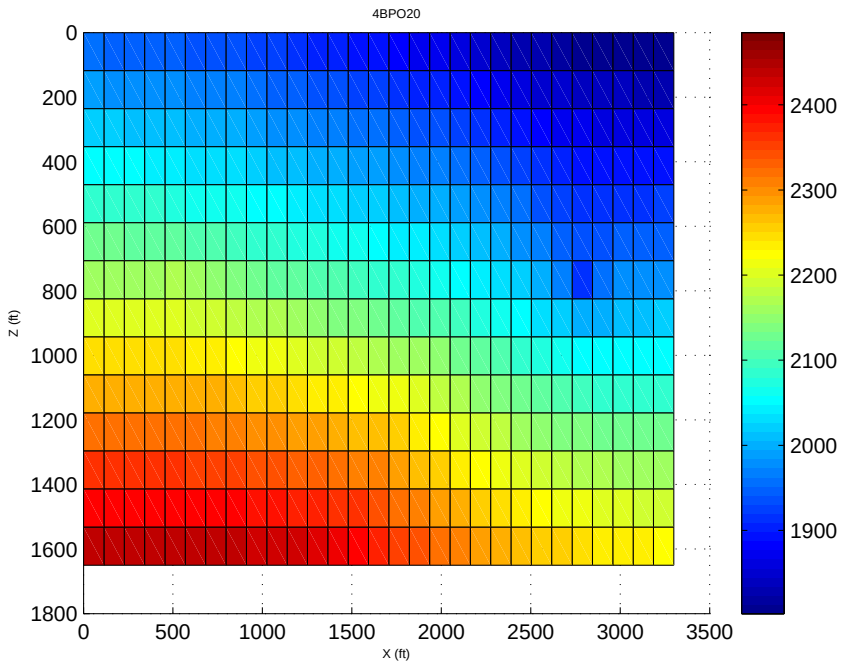

Figure 18: Oil pressure after 10-year IMPES simulation – test 4BIMPESLONG.

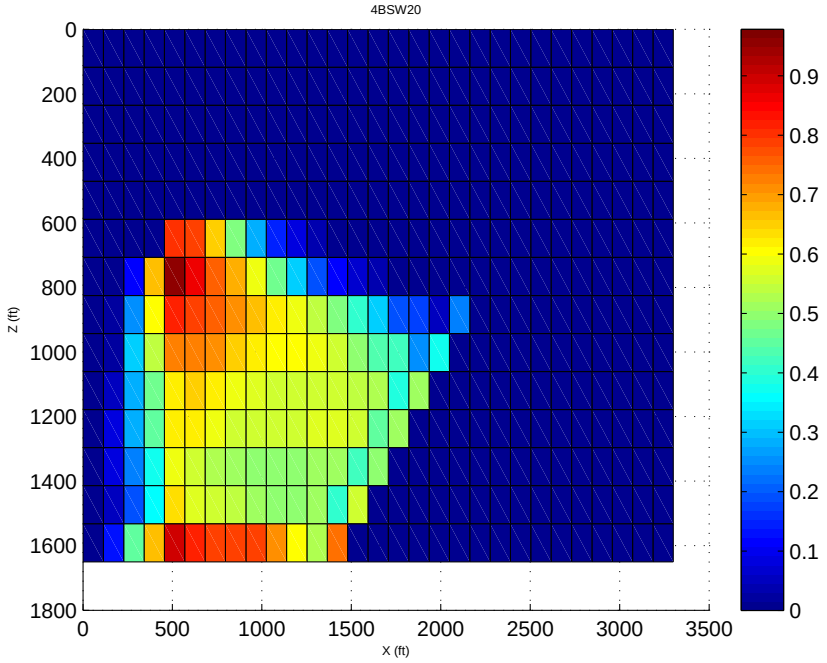

Figure 19: Water saturation after 10-year IMPES simulation – test 4BIMPES-LONG.

water saturation profiles at the end of the 5-year simulation by `OOMP_RS` are shown on Fig 20,21. Fig 22 indicates a slight increase in the production rate, and Fig 23,24 indicate an increase of the cumulative production in 5 years by $\approx 27,000$ `bbl`.

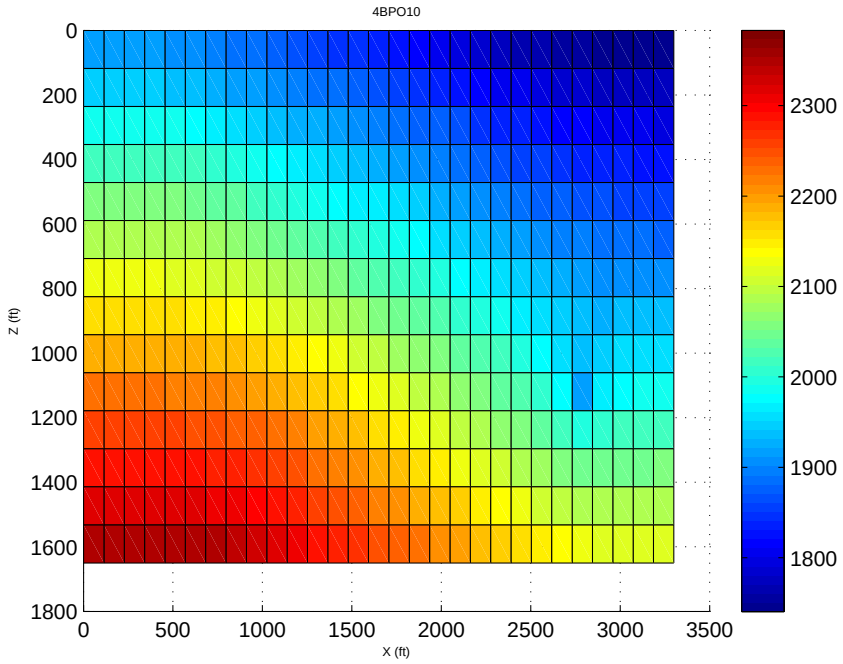

Figure 20: Oil pressure after 5-year FIM simulation – changed well locations – test 4BFIMMOVED.

11.2 Sensitivity to Mobility

In the sensitivity to mobility analysis, we increase the oil viscosity by factor of 2, thus decreasing its mobility by half. The resulting oil pressure and water saturation profiles are shown on Fig25,26. The predicted bottom-hole pressure drop at the producer (see Fig 27) is in agreement with the reduced oil mobility, and the corresponding production drop ($\approx 10,000$ `bbl`) is demonstrated on Fig 28,29.

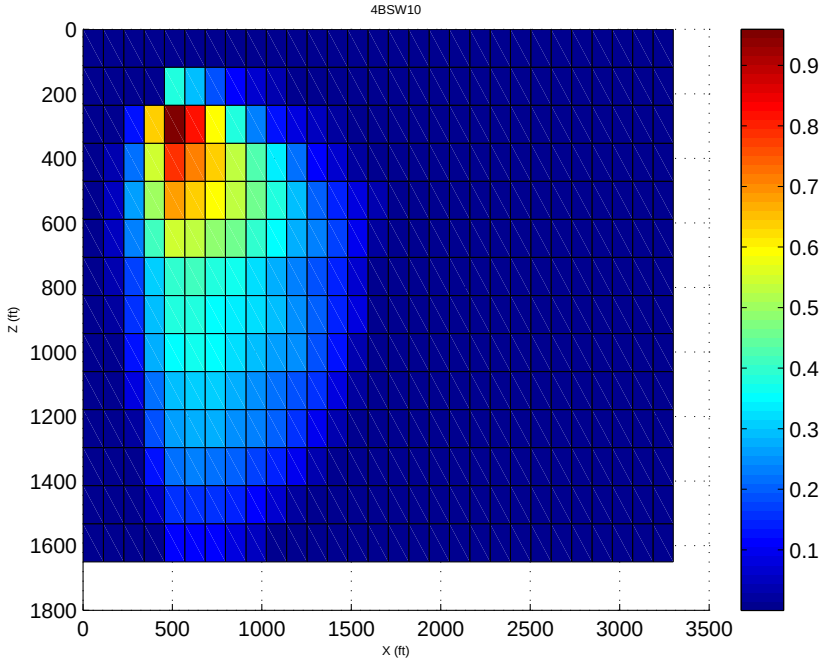

Figure 21: Water saturation after 5-year FIM simulation – changed well locations – test 4BFIMMOVED.

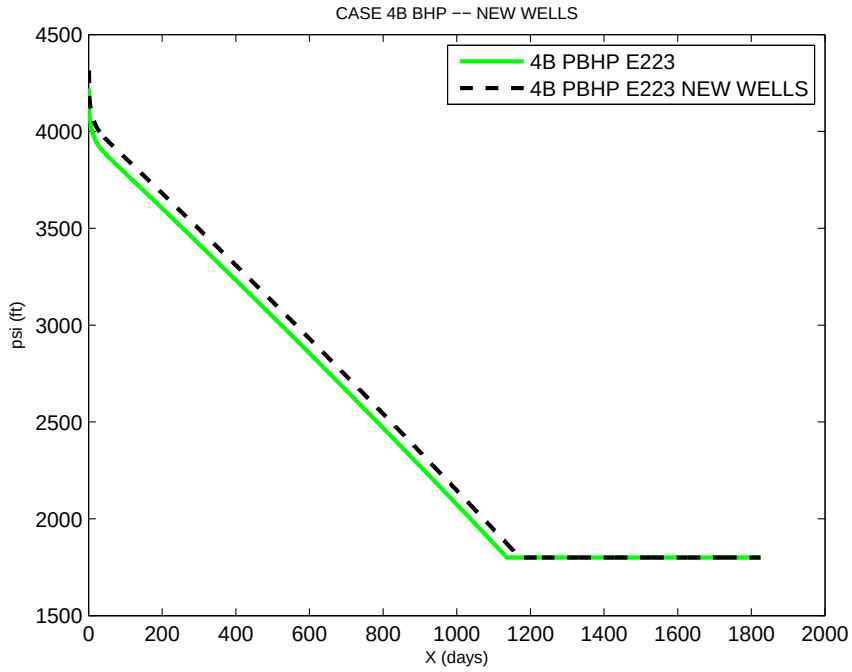

Figure 22: Well BHP – changed well locations – test 4BFIM vs 4BFIMMOVED.

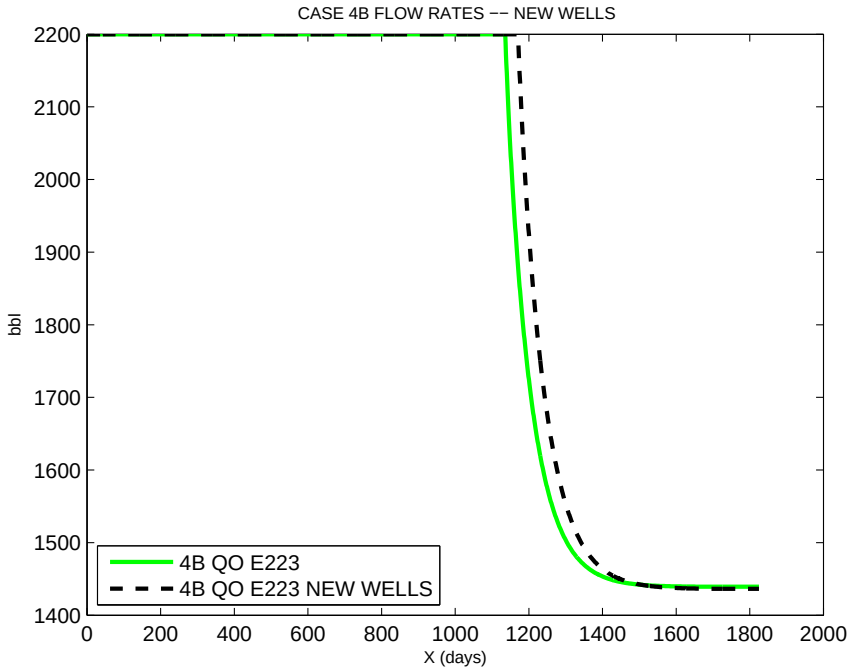

Figure 23: Production rates – changed well locations – test 4BFIM vs 4BFIM-MOVED.

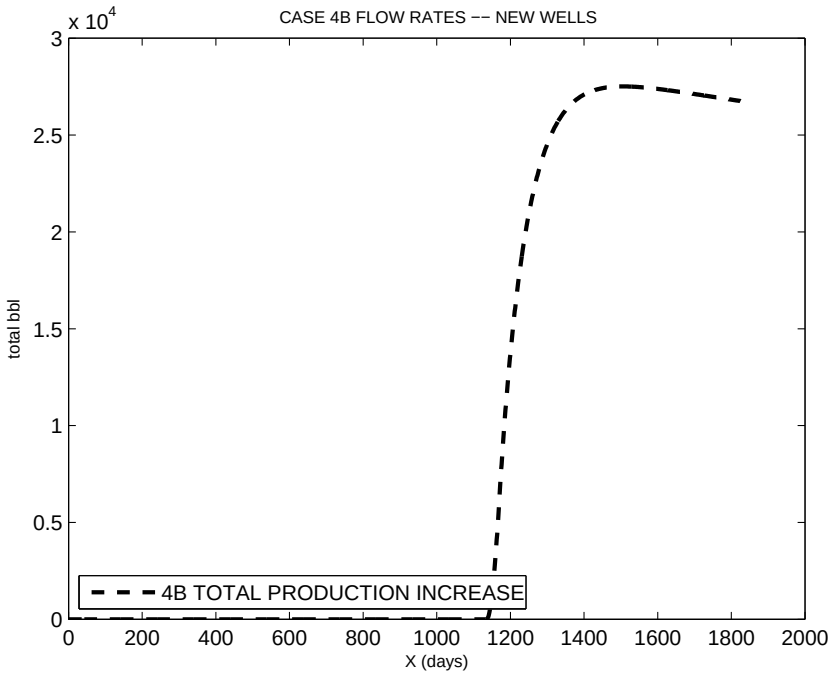

Figure 24: Production increase due to changed well locations – test 4BFIM vs 4BFIMMOVED.

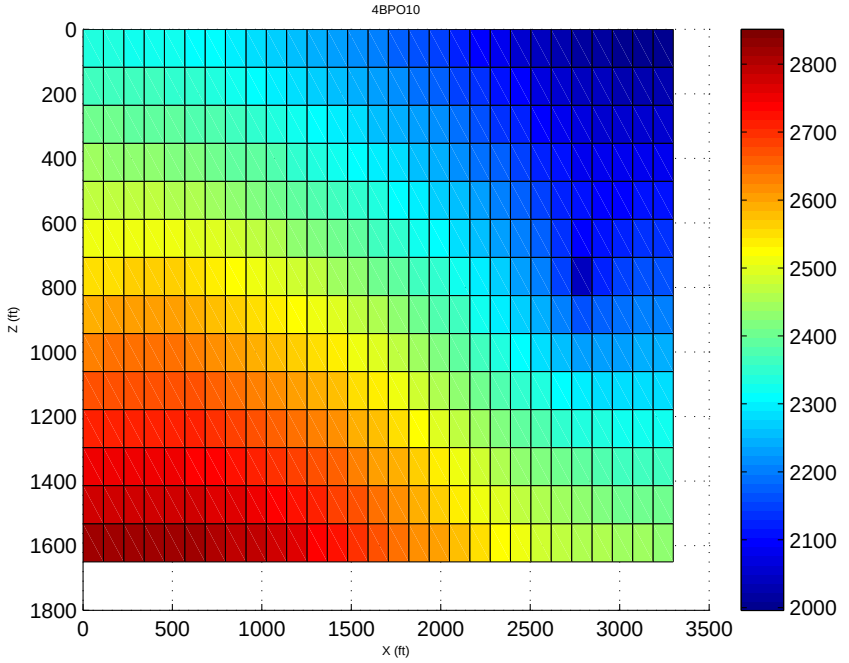

Figure 25: Oil pressure after 5-year FIM simulation – reduced oil mobility – test 4BFIMMOBILITY.

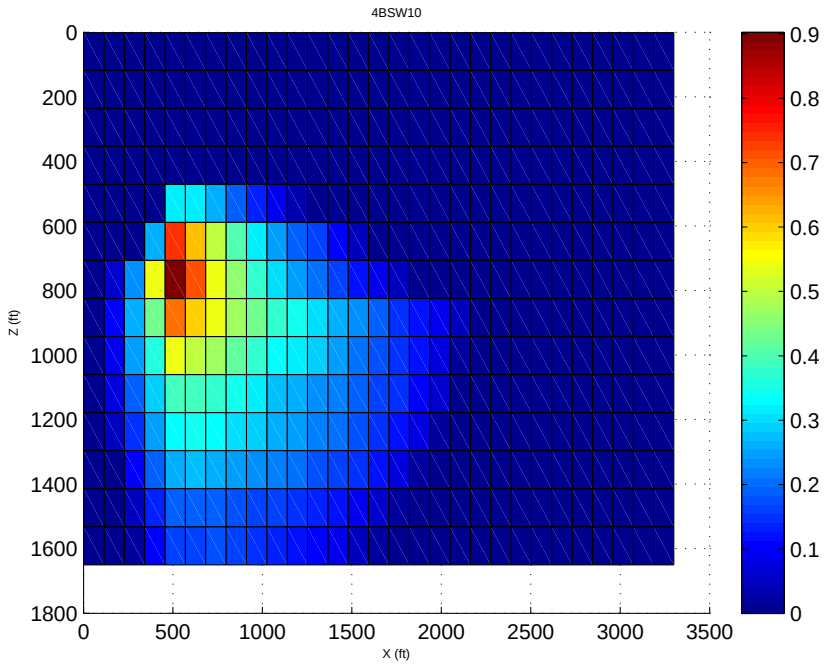

Figure 26: Water saturation after 5-year FIM simulation – reduced oil mobility – test 4BFIMMOBILITY.

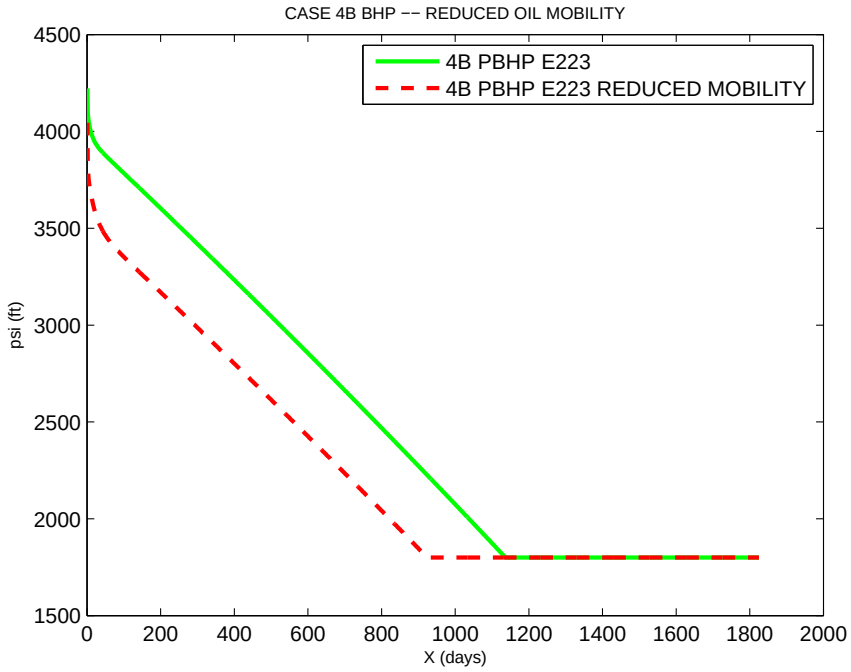

Figure 27: Well BHP – reduced oil mobility – test 4BFIM vs 4BFIMMOBILITY.

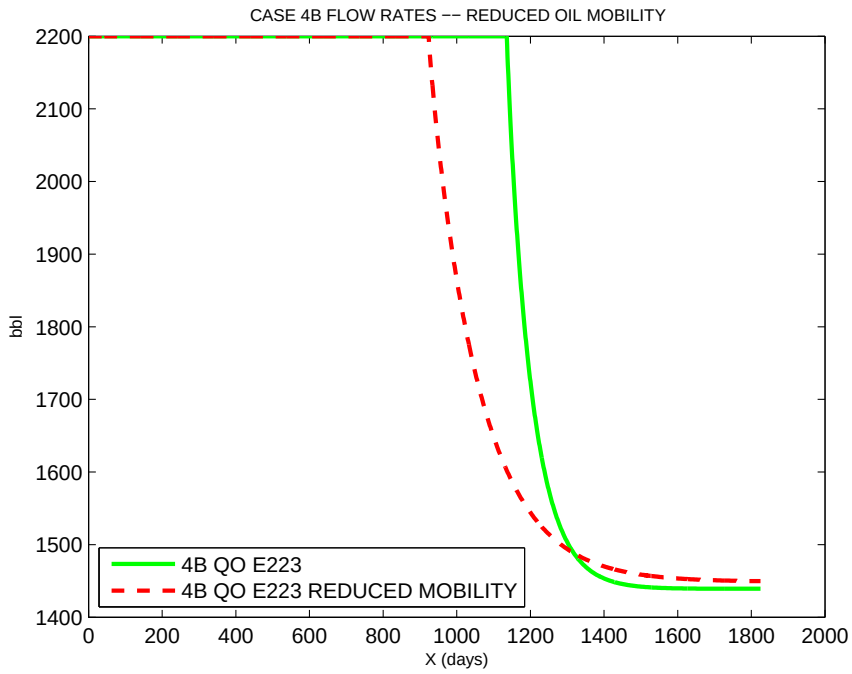

Figure 28: Production rates – reduced oil mobility – test 4BFIM vs 4BFIMMOBILITY.

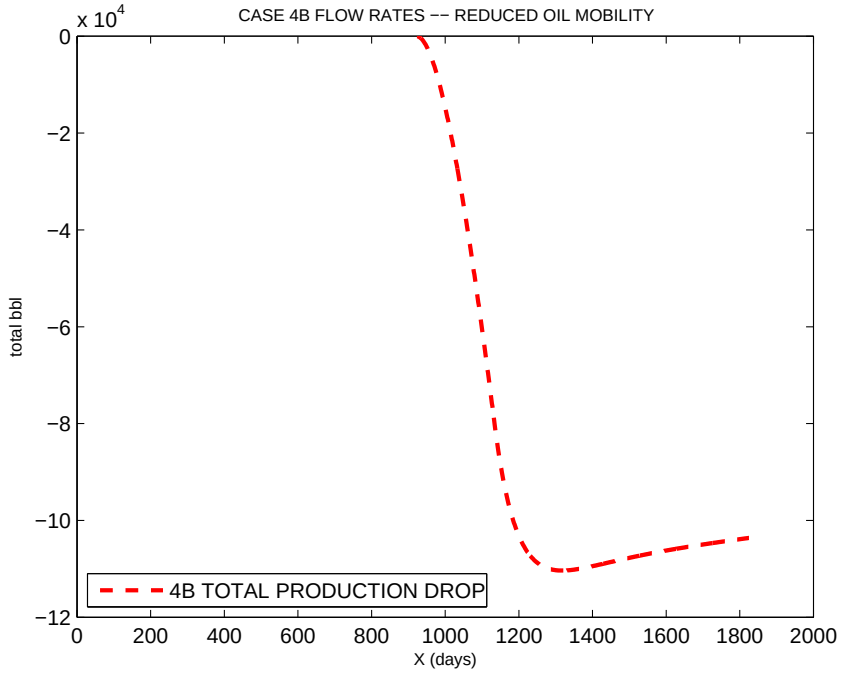

Figure 29: Production drop due to reduced oil mobility – test 4BFIM vs 4BFIM-MOBILITY.

11.3 Grid Refinement Study

In this study we increase the spatial resolution by a factor of two (Fig 3031)[19] and factor of 4(Fig 3334) [20]. Note that for *vertical cross sections* doubling the spatial resolution is not sufficient for increasing the accuracy because of the semi-analytical *well index* term (27) that depends *linearly on the vertical grid step*[21]. The reason for that is the (almost) linear increase of the well index would result in a horizontal shift of the BHP curve inconsistent with the physical model. Top avoid this phenomenon and in order to be able to compare the modeled reservoirs at varying resolution, we *manually specify the well index* in the high-resolution tests, using the well index values computed in the base low-resolution test.

Note a very good agreement between the bottom hole pressures for two different resolutions (see Fig32).

The highest resolution (120×60) test on Fig 33,34 is a good illustration of the "high permeability channel" that manifested itself earlier by a sharp horizontal contrast in water saturation, in the IMPES test of Fig19. The contour plot of Fig 35 shows a detailed structures of saturation fronts caused by the complex rock permeability patterns.

12 Time Step Control

`OOMP_RS` uses two principal time-step and iteration control methods.

1. for the FIM, the time step is controlled using a user-specified *desired change* in oil pressure and water saturation ([2]):

$$\Delta t^{n+1} \;=\; \frac{(1+\omega)\delta p^d}{\delta p + \omega \delta p^d}\Delta t^n, \tag{30}$$

   where δp^d is the desired change in variables p and δp is the actual change over the previous step. We combine (30) with a *dynamic step control* that cuts the time step by half if the Newton method fails to converge within a specified number of iterations;

2. for the IMPES, we use the CFL for an *approximation* to the saturation front propagation equation and compute the maximum allowed time step as

$$\Delta t \;=\; 0.5 \times \min \frac{V_{ijk}\phi_{ijk}}{(q_o + q_w)f'_w}, \tag{31}$$

   where f_w is the ratio of the water mobility to the total mobility of oil and water, and the coefficient 0.5 in (31) we have found heuristically to account for the approximate nature of the corresponding nonlinear first-order hyperbolic PDE ([1]). `OOMP_RS` combines (31) with an automated fall-back mechanism in case the saturation takes non-physical values (i.e. above one or below zero).

[19] increasing the total number of grid blocks by 4

[20] increasing the number of grid blocks by 16

[21] For horizontal cross-sections, increasing the spatial resolution while still using the well index formula (27) is still possible

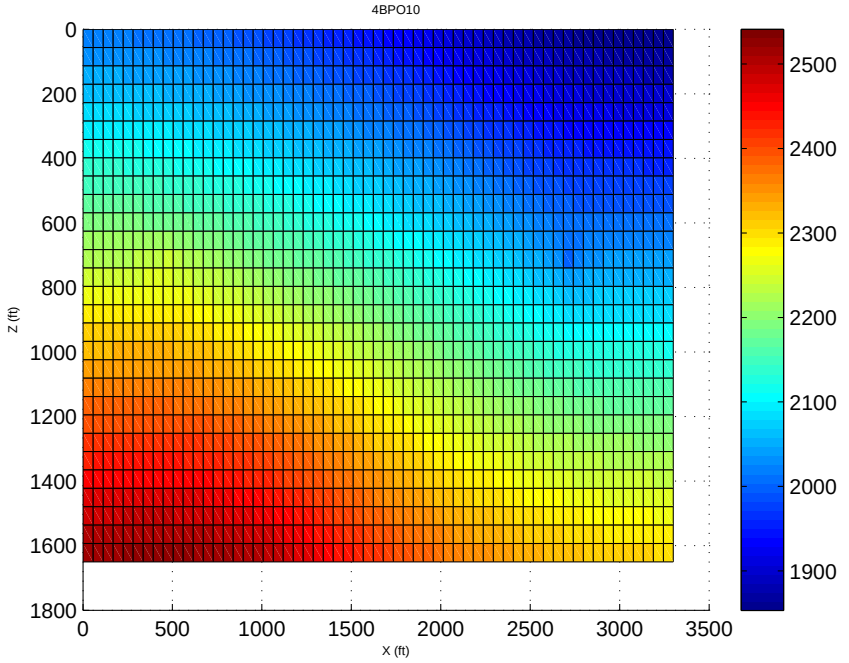

Figure 30: Oil pressure after 5-year FIM simulation – 2x resolution spatial grid – test 4BFIMHIRES.

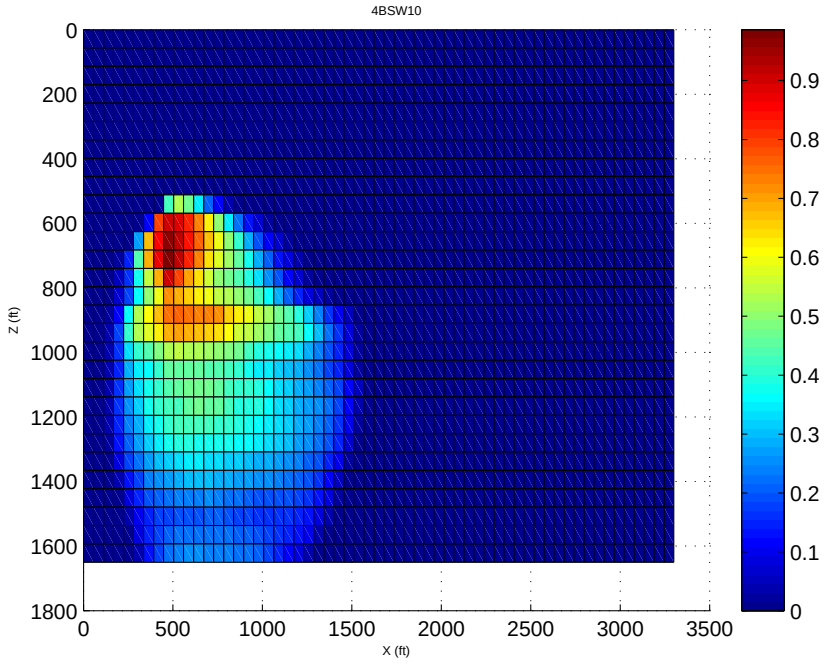

Figure 31: Water saturation after 5-year FIM simulation – 2x resolution spatial grid – test 4BFIMHIRES.

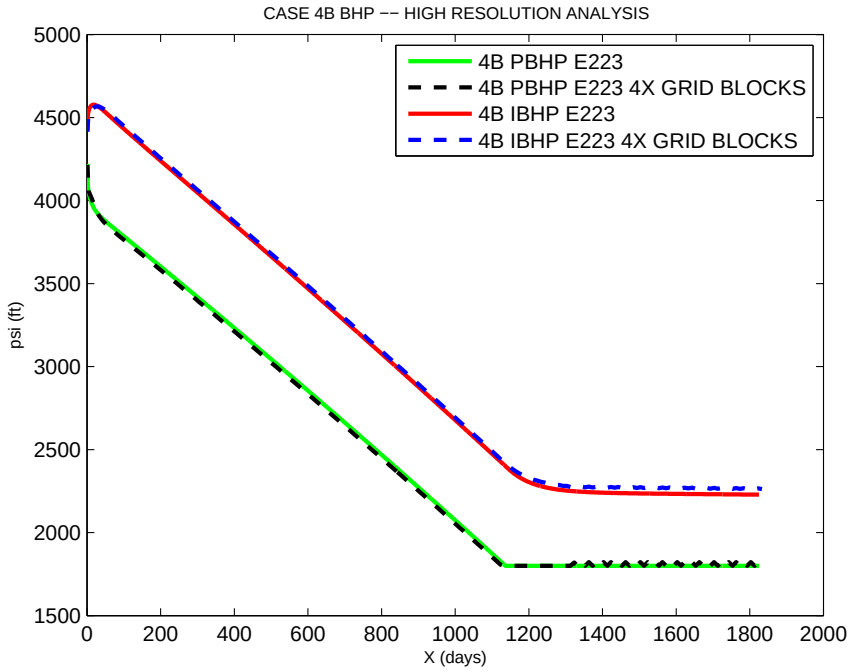

Figure 32: Well BHP – 2x resolution spatial grid – test 4BFIM vs 4BFIMHIRES.

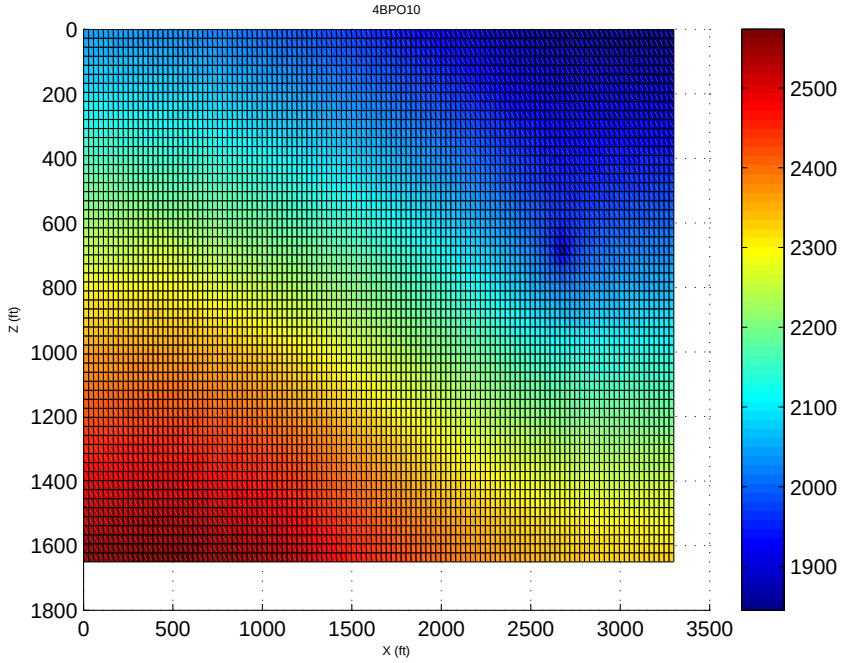

Figure 33: Oil pressure after 5-year FIM simulation – 4x resolution spatial grid – test 4BFIMHIRES2.

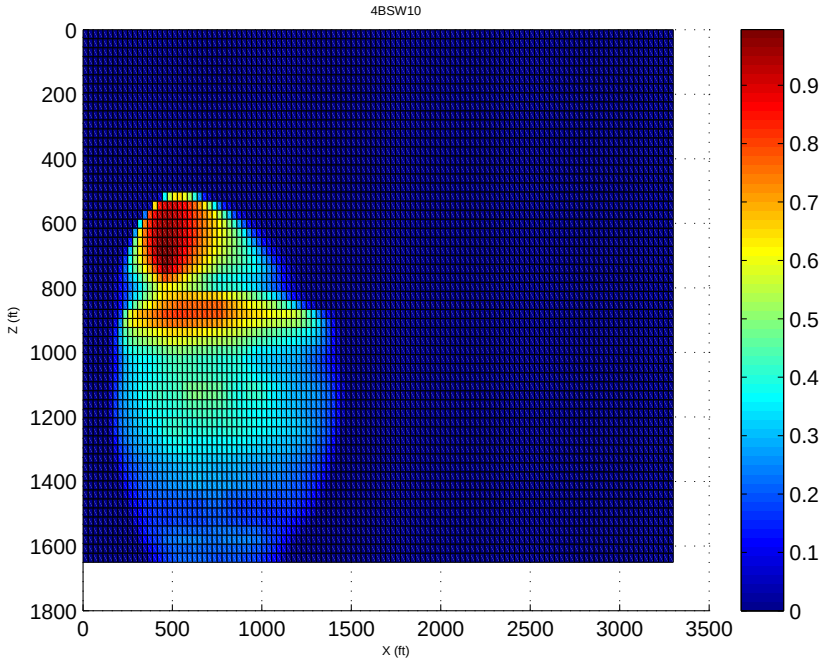

Figure 34: Water saturation after 5-year FIM simulation – 4x resolution spatial grid – test 4BFIMHIRES.

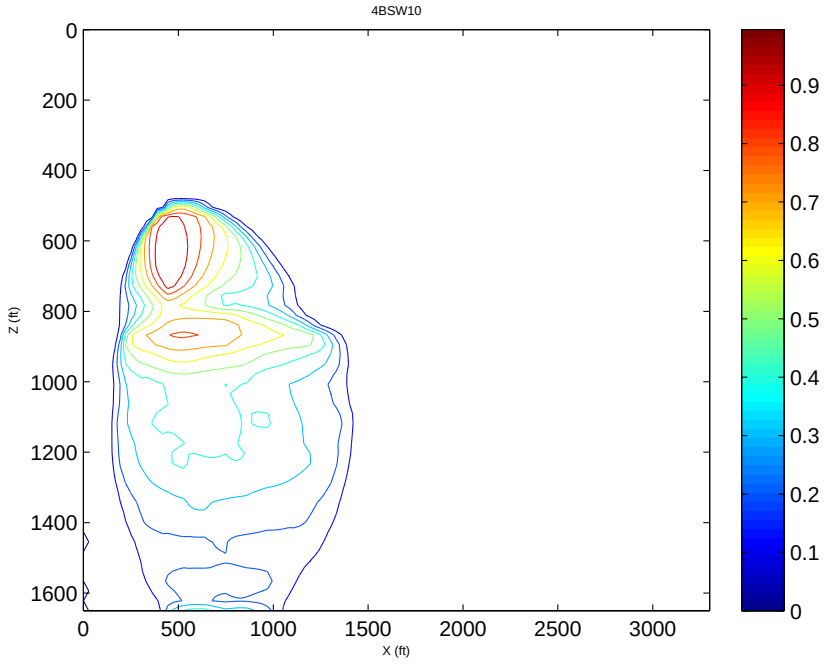

Figure 35: Water saturation after 5-year FIM simulation – 4x resolution spatial grid – test 4BFIMHIRES.

It is important to note that the automatic step control used in our simulator does not use any accuracy criteria beyond the maximum time step specified by the user.

Figure36,37 illustrate the time-step strategy used by three favours of the FIM and IMPES. The most accurate FIM uses a constant time step of 1 day and steadily decreasing number of Newton iterations. The fastest FIM (12 `s` run time *including file logging*) on an 8-core intel64 system) increases the time step to up to 50 days, while using more iterations. The second FIM demonstrates a trade-off between the time steps and iterations.

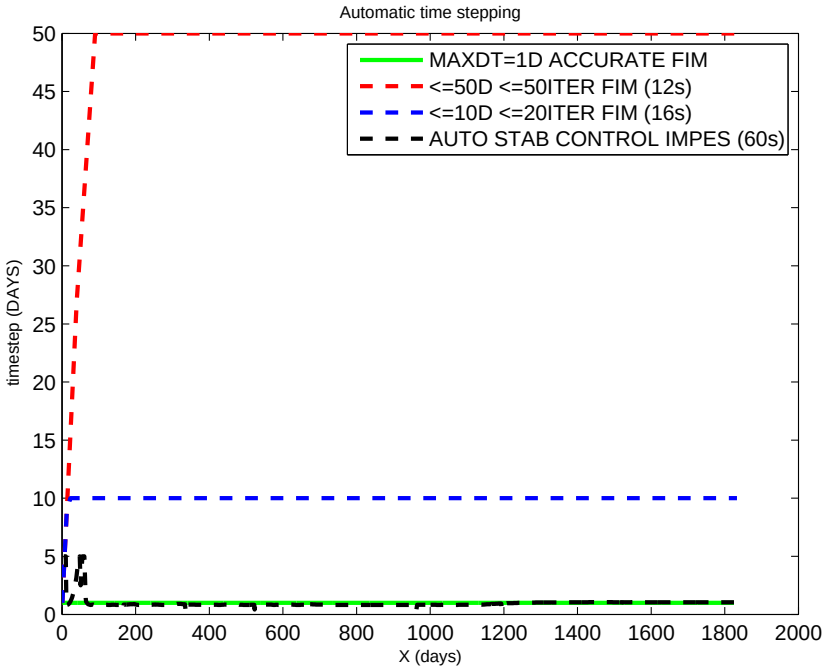

Figure 36: Comparison of time steps used by three FIM strategies and automatic stability control IMPES.

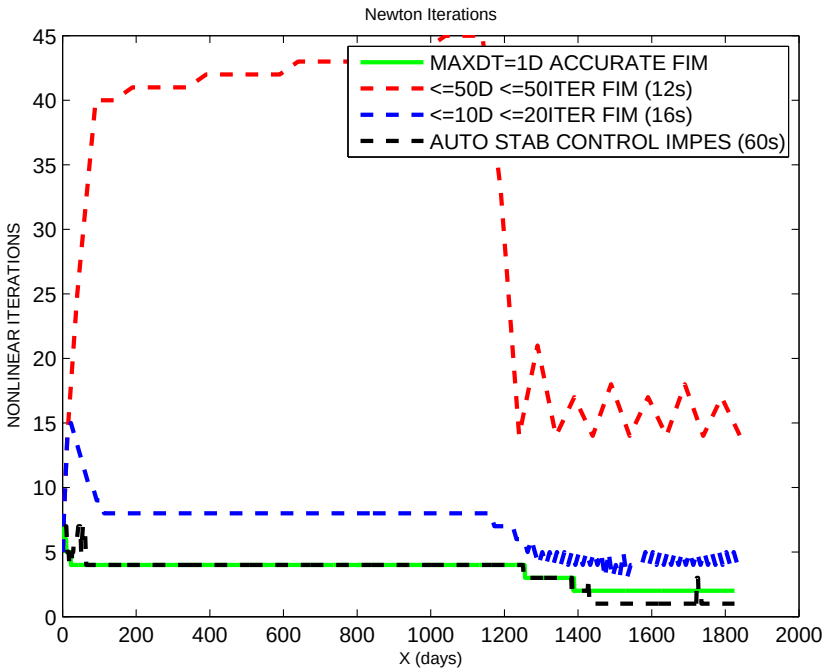

Figure 37: Comparison Newton iteration count used by three FIM strategies and automatic stability control IMPES.

13 Appendix A: Building the Code

The simulator code resides in the `src` subfolder of the report folder. The Bourne shell script `compile.sh` can be used to compile and build the simulator on an `intel64 (emt64)` platform running Linux (kernel 2.6.18 or later) and `Intel Fortran Compiler` version 12.0 or later, and `Intel Math Kernel Library` version 10.2 or later.

If `Intel` development environment has been set up for your account, then you should be able to build the simulator simply by running the above script. If the code builds but fails to run due to a missing dynamic library, check your `$LD_LIBRARY_PATH` variable and make sure that paths to the `Intel` run-time libraries are included in the variable[22].

All of the tests described in this document reside in the `tests` subfolder of the report root. Change the current directory to any one of those directories and run the corresponding tests by invoking the `run.sh` script. The program output will appear in text files within the same folder and (partially) on the console screen. Small`Matlab` scripts are provided inside the test folders for generating the intermediate and final modeling plots.

The parent directory `tests/` contains a `Matlab` script that generates the bottom-hole pressure, production and time-stepping plots.

14 Appendix B: Data File Format

The following is the sample data input file used by version 2.0 of `OOMP_RS`. The variable names are self-explanatory and/or agree with the definitions of the Phase 4 task formulation.

```
!
! PHASE 4B parameters
!
&dimensions lx=3300,ly=110,lz=1650,topdepth=6000. /

&permeabiulity fnpermx='rockperm4b.inc', fnpermy='rockperm4b.inc', fnpermz='rockperm4b.inc'

&porosity fnpor='porosity.inc', fncompr='rockcompr.inc', pref_por=14.7 /

&tops toppressure=4200. /

&gridsize nx=30, ny=1, nz=15 /

&name casename='4B' /

&oildensity rho0_o=40., b0_o=1., cf_o=1.2e-5, pref_rho_o=14.7 /

&waterdensity rho0_w=62.238, b0_w=1., cf_w=5.e-7, pref_rho_w=14.7 /

&watermu mu0_w=1., cmu_w=0., pref_mu_w=14.7 /
```

[22] or specified in ld.conf

```
&oilmu mu0_o=5., cmu_o=2.e-6, pref_mu_o=14.7 /

&capillary pc_c=0., pc_exp=2. /

&fluidpermeability kro_exp=1.5, krw_exp=1.5 /

&injector iw_ix=5, iw_iy=1, iw_iz=7, iw_qw=-1400, iw_maxbhp=7500, iw_radius=0.5 /

&producer pw_ix=25, pw_iy=1, pw_iz=7, pw_qo=2200, pw_minbhp=1800, pw_radius=0.5 /

&control tol=0.0001, minstep=.1, maxstep=1, initstep=1, maxtime=1825, maxnewton=50 /

&stepcontrol omega=.5, target_p=200., target_s=.1 /

&output dout=182, welldout=1, interm=1, stepout=1 /

&method impes=0 /
```

References

[1] Uri Ascher. *Numerical Methods for Evolutionary Differential Equations*. SIAM, 2008.

[2] Khalid Aziz and Antonin Settari. *Petroleum Reservoir Simulation*. Stanford Course Supplement, 2002.

[3] V. A. Barker, L. S. Blackford, J. Dongarra, J. Du Croz, S. Hammarling, M. Marinova, J. Wasniewski, and P. Yalamov. *LAPACK95 user's guide*. SIAM, 2001.

[4] Zhangxin Chen. *Reservoir Simulation: Mathematical Techniques in Oil Recovery*. SIAM, 2007.

[5] Jon Claerbout. *EARTH SOUNDINGS ANALYSIS: processing versus Inversion*. SEP, 2004. http://sepwww.stanford.edu/sep/prof/pvi.pdf.

[6] Gene H. Golub and Charles F. van Loan. *Matrix Computations*. Johns Hopkins University Press, 1996.

[7] Willem Hundsdorfer and Jan G. Verwer. *Numerical Solution of Time-Dependent Advection-Diffusion-Reaction Equations*. Springer-Verlag, 2010.

[8] Michael Metcalf, John Reid, and Malcolm Cohen. *Fortran 95/2003 explained*. Oxford University Press, 2004.

[9] Andrzej Ruszczynski. *Nonlinear Optimization*. Princeton University Press, 2006.

[10] Lloyd Trefethen and David Bau. *Numerical Linear Algebra*. SIAM, 1997.