Abstract
Reservoir simulators that efficiently calculate gradients are not yet widely used, because for few control parameters (or little data available), it is feasible to approximate gradients by numerical perturbations. As more parameters need to be handled, the need for efficiently calculating gradients increases.
In this paper it is shown how to implement a Black-Oil reservoir simulator for (under)saturated oil with gradient capabilities.
The usefulness of implementing reservoir simulators with Jacobian capabilities is shown by embedding reservoir simulation in a closed loop setting. A gradient-based optimization routine can determine how inflow control valves in smart wells should be handled in order to optimize oil recovery or net present value. Moreover a gradient-based data assimilation routine improves the reservoir simulator's predictive capabilities by updating parameters every time new measurements become available.

Introduction
With the improvement of smart well hardware, the reservoir engineer can, to a certain extent, steer the flow through the reservoir by indirectly imposing a desired pressure gradient to the reservoir. The reservoir engineer has direct control over the flow rate (in rate constrained well operation) or the wellbore pressure at the surface (in pressure constrained operation) and the apertures of the inflow control valves (ICV's) in the production and injection wells.
The smart wells must of course be operated smartly in order to earn back the extra investments that have to be made in the early stages of the lifetime of the reservoir. (Sub-)optimal strategies are devised by means of reservoir simulators. The simplest way is running the simulator with many different inputs to see which one results in the highest net present value (NPV). A more advanced method is by systematically searching through "control space" by finding a direction (gradient) that best improves the current found strategy. Such a gradient can be found by perturbing the current strategy to see which perturbation improves the NPV the most. As the number of control parameters increases, more perturbations are needed and gradient-based routines, like optimization routines, that run on top of reservoir simulators become infeasible. It is possible to implement reservoir simulators that can predict future reservoir states based on current states and also give the derivatives of the future states with respect to the current states at little extra calculation time. Implementing such a simulator can be quite cumbersome though, especially when the simulator is set up to deal with complicated physical phenomena.

Closed Loop Reservoir Simulation
The term closed loop reservoir management [5] is used when a data assimilation routine is present in addition to a reservoir simulator and an optimization routine (Fig. 1). Traditionally the parameters of the reservoir simulator are history matched a few times during the lifetime of the reservoir. However by integrating sensors in the smart wells, it has become possible to have data available on a weekly or daily basis. When formulated in a general way, data assimilation algorithms provide a framework in which data from varying different sources, like sensors in the smart wells or seismics, can together contribute to updating model parameters of different natures of uncertainty, like permeability, porosity or PVT data. "The loop is closed" when data assimilation routines are put in place to provide nearly continuous reservoir monitoring. From a systems engineering point of view, there are actually two closed loops when both an optimization routine and a data assimilation routine are manipulating the output of the reservoir simulator.
A reservoir simulator can be seen as a mechanism that predicts the future state of the reservoir given the current state and the user input:

$$x_{k_t} = f(x_{k_t}, u_{k_t})$$

(1.1)

or if \( f \) is not known explicitly, then \( x_{k_t} \) must be solved implicitly from

$$f(x_{k_t}, x_{k_t-1}, u_{k_t}) = 0$$

(1.2)

The state of the reservoir \( x \) can for example consist of values for the pressure and saturation of the fluid phases in all grid blocks. These values can be combined together by POD or other reduction algorithms [6]. Moreover, the state variables might not even represent any physical property; they might be parameters in some neural network that mimics the input-output behavior of the reservoir.

**Gradient-based Production Optimization**

The objective function is defined on a time window and translates the reservoir state into some objective. The time window of interest might for example be the total lifetime of the reservoir and the objective might be total recovered oil or NPV.

$$J = \sum_{i=t}^{N} J_i(x_{k_t}, u_{k_t})$$

(1.3)

where \( t \) are the times at which the control parameters are changed. If the control parameters were to be perturbed, the dynamics of the reservoir states would also change. Or, if in addition the dynamics were kept unchanged, then Eq. (1.2) would no longer be satisfied. Therefore the dynamics are added to the objective function as a penalty

$$H_i(x_{k_t}, x_{k_t-1}, u_{k_t}, \lambda_{k_t}) = J_i(x_{k_t}, u_{k_t}) + \lambda_{k_t}^r f(x_{k_t}, x_{k_t-1}, u_{k_t})$$

(1.4)

$$H(x_{k_t}, x_{k_t+1}, x_{k_t+2}, \ldots, u_{k_t}, \ldots, u_{k_{T-1}}) = \sum_{i=t}^{N} H_i(x_{k_t}, x_{k_t-1}, u_{k_t}, \lambda_{k_t})$$

(1.5)

This is basically a trick to decouple the dependence of \( u_{k_t} \) and \( x_{k_t} \), or it can be seen as applying the chain rule for differentiating compositions of functions.

It can be shown [4] that the (local) optima of \( J \) occur in the saddle points of \( H \), provided that they exist. They can be found by setting

$$\frac{\partial H}{\partial \lambda_{k_t}} = 0 \quad \forall k \in \{0, \ldots, N\}$$

(1.6)

$$\frac{\partial H}{\partial x_{k_t}} = 0 \quad \forall k \in \{0, \ldots, N\}$$

(1.7)

$$\frac{\partial H}{\partial u_{k_t}} = 0 \quad \forall k \in \{0, \ldots, N\}$$

(1.8)

Equation Eq. (1.6) leads back to Eq. (1.2), which means that only one forward simulation is needed to calculate the dynamics of the reservoir state over the control window, instead of running as many forward simulations as there are control parameters. Eq. (1.7) can be written as

$$\frac{\partial J_i(x_{k_t}, u_{k_t})}{\partial x_{k_t}} + \frac{\partial}{\partial x_{k_t}} \left( \lambda_{k_t}^r f(x_{k_t}, x_{k_t-1}, u_{k_t}) + \lambda_{k_t}^r f(x_{k_t}, x_{k_t-1}, u_{k_t}) \right) = 0$$

(1.9)
Starting with $\lambda_{0} = 0$, the so called Lagrange multipliers $\lambda$ can be calculated by a backward simulation with the transpose of the locally linearized reservoir simulator

$$\lambda_{i} = - \frac{\partial f(x_{n}, x_{n}, u_{n})}{\partial x_{n}} - \left( \frac{\partial f(x_{n}, x_{n}, u_{n})}{\partial x_{n}} \right)^{T} \lambda_{n}.$$ (1.10)

This is called the adjoint reservoir simulator. After running the simulator to calculate the reservoir state dynamics, the adjoint simulator is run to obtain the Lagrange multipliers. The Lagrange multipliers are used to build the gradient of the objective function with respect to the control parameters. If Eq. (1.8) is satisfied, then the optimal control is found. If not, the gradient is used to update the current strategy and the process is repeated. The computation time of one update is in the order of the computation time of three simulations: one forward simulation, one backward simulation and some overhead. This is dramatically lower than doing an update by numerical perturbations. In that case the number of simulations per update is equal to the number of control variables (times the number of time steps at which the controls can be changed).

**Gradient-based Data Assimilation**

Gradient-based data assimilation is basically no different than production optimization. In production optimization the control parameters are optimized with respect to e.g. NPV. In data assimilation the uncertain rock or fluid parameters of the reservoir simulator are optimized with respect to the difference between the real measurements and the measurements that the simulator predicts. If all uncertain parameters are placed in the vector $\alpha$, then the objective function for the data assimilation routine looks exactly like Eq. (1.3)

$$J = \sum_{j=0}^{\lambda_{i}} J_{i} \left( x_{n}, \alpha_{n} \right)$$ (1.11)

where $t_{j}$ are the times at which measurements are available. Again, the forward and backward simulations are run and the results are used to calculate the gradient $\frac{\partial H}{\partial \alpha_{n}}$.

Sometimes not the parameters are updated, but the initial condition. In that case, $\lambda_{n}$ is not calculated by setting $\frac{\partial H}{\partial x_{n}} = 0$, but $\frac{\partial H}{\partial \alpha_{n}}$ is calculated by setting $\lambda_{n} = 0$.

**Study Case: Optimal Strategy for Saturated and Undersaturated Oil**

This study case is meant to illustrate that dynamic control of a waterflood using smart well technology can easily lead to significant pressure fluctuations in the reservoir. The pressure fluctuations often result in the situation where the amount of gas dissolved in the oil is not simply a function of pressure anymore.

A 3D horizontal reservoir is considered. The reservoir has two high permeability streaks at either side perpendicular to a horizontal injector and producer (Fig. 2) and is otherwise homogenous. The water injector is completed in the bottom layer of the reservoir whereas the producer is situated in the top layer. Moreover both wells can be separated into two independent sections (a and b) by means of a smart well completion. The reservoir initially is at bubble point. In the initial stages the average reservoir pressure is dropped as the water injectors are pressure constrained close to the initial pressure. In the conventional case the pressure drops monotonically throughout the reservoir. Consequently gas will come out of solution. The smart well completion, when operated intelligently, can improve the waterflood recovery as shown in Fig. 3 where the conventional case (dashed lines) (i.e. no well section separation) is compared with a smart well scenario (solid lines) based on Brouwer\footnote{European Conference on the Mathematics of Oil Recovery — Cannes, France, 30 August - 2 September 2004}. However the dynamic control results in pressure fluctuations (Fig. 4). As a result the oil may become undersaturated. The incorrect pressure and undersaturation (difference between saturated gas-oil ratio and actual value) for one of the grid blocks. As the pressure goes down, initially saturated oil will remain saturated. However as the pressure goes up the amount of gas that can dissolve in the oil increases. If insufficient gas is available, the oil will become undersaturated. As subsequently the pressure goes down again the degree of undersaturation will decrease. The incorrectly added gas will come out of solution and create a three phase state that in reality would not occur.

\footnote{Note that the smart well strategy used, although a significant improvement over the conventional case, is not the optimal strategy.}
Figure 3: The rate and production profile for the conventional (dashed curves) and smart well (solid curves) case. Below the graph the valve setting is displayed (solid = open).

Figure 2: Top view of reservoir, indicating high permeability zones (dark gray) and well positions

Figure 4: Difference between saturated and actual gas-oil ratio.

Undersaturated Reservoir Simulation

A reservoir simulator predicts the future state of the reservoir (pressure and saturation of all fluid phases) given the current state and the user input, which can for example consist of flow rates in the wells (for rate constrained operation) or the wellbore pressure at the surface (for pressure constrained operation) along with the apertures of the inflow control valves. It typically implements mass balance equations for the (pseudo-)components and Darcy’s law applied to the individual phases (Aziz, et al.[1]):

\[
\frac{\partial}{\partial t} \left( \phi a \phi b S_a \right) - \nabla \cdot \left( a \phi b \nabla p - \rho_a g V D \right) = q_A \alpha
\]  

(1.12)

\[
\frac{\partial}{\partial t} \left( \phi a b S_b \right) - \nabla \cdot \left( a \phi b \nabla p - \rho_b g V D \right) = q_A \alpha
\]  

(1.13)

\[
\frac{\partial}{\partial t} \left[ \phi a b \left( b_S + R b_S \right) \right] - \nabla \cdot \left( a \phi b \nabla p - \rho_a g V D \right) - \nabla \cdot \left( a \phi b \nabla \left( p_S - \rho_a g V D \right) \right) = \left( q_g + q_r \right) \alpha
\]  

(1.14)

Here \( p_a, S_a, b_a, \mu_a, \rho_a \) and \( \lambda_a = k_a s / \mu_a \) are the pressure, saturation, reciprocal formation volume factor, viscosity, density and mobility of phase \( a \), \( k_a \) is the permeability of the rock for single phase flow, \( k_{r a} \) is the relative permeability of phase \( a \) in the presence of other phases, \( R \) is the gas-oil ratio, \( D \) is the depth, \( g \) is the gravitational acceleration constant, \( q_g \) is the injection(+) or production(-) density of phase \( a \) expressed in volumes at standard conditions and \( \alpha \) is a geometric factor.

For saturated oil the pressure of one phase and the saturations of two phases can be chosen as the minimal representation of the reservoir state, for example \( x = \{ p_a, S_a, S_b \} \). The other pressures and saturation and properties can then be derived from the reservoir state:

\[
S_e = 1 - S_a - S_b
\]  

(1.15)

\[
p_s = p_a - p_{sw} \left( S_a \right), \quad p_p = p_a + p_{ov} \left( S_b \right)
\]  

(1.16)

\[
b_s = b_a \left( p_a \right), \quad \mu_a = \mu_a \left( p_a \right)
\]  

(1.17)
\[ R = R^{\text{sat}}(p_s) \]  
(1.18)

\[ \rho_o = b_o \rho_s^{\text{STC}}, \quad \rho_g = b_g (\rho_s^{\text{STC}} + \rho_p^{\text{STC}}), \quad \rho_p = b_p \rho_s^{\text{STC}} \]  
(1.19)

\[ k_{oo} = k_{oo}(S_o, S_g) \]  
(1.20)

For undersaturated oil Eq. (1.18) is no longer valid. Instead \( R < R^{\text{sat}}(p_s) \). As minimal state representation of an undersaturated reservoir, \( x = \{ p_s, S_o, R \} \) can be chosen. Equations Eq. (1.15) and Eq. (1.18) must be replaced with

\[ S_o = 0 \]  
(1.21)

\[ S_g = 1 - S_o \]  
(1.22)

The bubble point pressure \( p_b \) is defined as the pressure of the oil phase at which gas component bubbles out of the oil phase. Mathematically it can also be defined as the pressure of the oil phase at which the switch between the saturated and undersaturated regimes would be made if the mass of the gas component is held constant and the oil phase pressure is altered, so equations Eq. (1.15), Eq. (1.18), Eq. (1.21) and Eq. (1.22) all hold. For the undersaturated regime \( (p_s > p_b) \), the user input functions \( R^{\text{sat}}(p_s) \) and \( b^{\text{sat}}_o(p_s) \) have to be modified

\[ R(p_s) = R^{\text{sat}}(p_s) \quad b_o(p_s) = b^{\text{sat}}_o(p_s) e^{-(p_b-p_s)} \]  
(1.23)

where \( c_o \) is the oil phase compressibility for undersaturated oil, a new user input. Optionally, also the oil viscosity \( \mu_o \) may be modified for the undersaturated regime.

The saturated and the undersaturated reservoir systems can be integrated using the state variable

\[ g = \left( \rho_s \rho_o \rho_g \right) \]  
(1.24)

if it can be detected how the gas component mass should be distributed over the oil and gas phase. This is typically a maximization problem: as much of the gas component mass is placed in the oil phase as is allowed by the oil pressure and the saturated oil-gas ratio \( R^{\text{sat}}(p_s) \). If there is still gas component mass left, it is placed in the gas phase. The gas component mass that is needed to switch between the saturated and undersaturated regimes can be calculated from Eq. (1.21) and Eq. (1.24) as

\[ g^{\text{swi}} = R^{\text{sat}}(p_s) \]  
(1.25)

In case \( m_g > m_g^{\text{swi}} \), the oil is saturated with gas, so there must be gas component gas phase. \( 1 - S_o \), must then be split into \( S_o \) and \( S_g \). This can be done by combining equations Eq. (1.15), Eq. (1.16) and Eq. (1.24) to calculate \( S_o = 1 - S_a - S_g \) after solving \( S_g \) from

\[ m_g = b_g \left( p_s + p_{w,s}(S_o) \right) S_g + R b_o \left( 1 - S_a - S_g \right) \]  
(1.26)

The system of non-linear coupled algebraic equations in \( x = \{ p_s, S_o, m_g \} \) that is obtained after discretization, using central differences for the spatial derivatives and Euler backwards for the time derivatives, looks like

\[ \frac{\phi a b S_o}{t_{i+1} - t_i} - V \cdot (\alpha \lambda^o \nabla p_s) + V \cdot (\alpha \lambda^g \nabla p_g) + gV (\alpha \lambda^o \rho_o) \cdot \nabla D - q_o \alpha \]  
(1.27)

\[ \frac{\phi a b S_o}{t_{i+1} - t_i} - V \cdot (\alpha \lambda^g \nabla p_g) + gV (\alpha \lambda^o \rho_o) \cdot \nabla D - q_o \alpha \]  
(1.28)

\[ \frac{\phi a m_g}{t_{i+1} - t_i} - V \cdot ((\alpha \lambda^o + \alpha \lambda^g R) \nabla p_\text{wo}) - V \cdot (\alpha \lambda^g \nabla p_g) + gV (\alpha \lambda^o \rho_o + \alpha \lambda^g \rho_g) \cdot \nabla D - (q_g + q_R) \alpha \]  
(1.29)

or

\[ \left( \Psi_s, s - T_p s + T_p \rho_o + g_s - q_s \right)_{t_{i+1}} = \left( \Psi_s, s \right)_{t_i} \]  
(1.30)

\[ -\Psi_s, s + \Psi_p, \left( 1 - s \right) - T_p s + g_s - q_s \]  
(1.31)

\[ \Psi g = T_p s - T_p \rho_o + g_s - q_s \]  
(1.32)

where \( \Psi \) are diagonal matrices and \( T \) are \((2d+1)\) -diagonal matrices ( \( d \) is the number of spatial dimensions).

Undersaturated reservoir simulation can schematically be summarized as shown in figure Fig. 5:
Given $x_{satur} = \{p_s, S_s, m_t\}_{satur}$

- $m_t > R'_{satur} h'_{satur} (1 - S_s)$ ?
  - Yes
    - Undersaturated
      - $S_s = 0\quad S_s = 1 - S_s$
      - $p_s = p_s - p_{row} (S_s)$
      - $b_s = b_s (p_s)\quad b_t = b_t (p_s)$
      - $R = R^{satur} (p_s)$
      - $\rho_s = \rho_{s,T}^{satur}$
      - $\mu_s = \mu_s (p_s)$
      - $k_s = k_s (S_s,S_s)$

    - $\frac{\partial}{\partial t} (\phi a b S_s) - \nabla \cdot (a \phi S_s p_s) + \nabla \cdot (a \phi S_s p_{row}) + g \nabla (a \phi S_s p_s) \cdot \nabla D - q \alpha$

    - $\frac{\partial}{\partial t} (\phi a b S_s) - \nabla \cdot (a \phi S_s p_s) + \nabla \cdot (a \phi S_s p_{row}) + g \nabla (a \phi S_s p_s) \cdot \nabla D + g \nabla (a \phi S_s p_{row}) \cdot \nabla D = (q_s + q_r) \alpha$

- No
  - Saturated
    - $S_s = 1 - S_s - S_s$
    - $p_s = p_s - p_{row} (S_s)$
    - $b_s = b_s (p_s)\quad b_t = b_t (p_s)$
    - $R = R^{satur} (p_s)$
    - $\rho_s = \rho_{s,T}^{satur}$
    - $\mu_s = \mu_s (p_s)$
    - $k_s = k_s (S_s,S_s)$

  - $\frac{\partial}{\partial t} (\phi a b S_s) - \nabla \cdot (a \phi S_s p_s) + \nabla \cdot (a \phi S_s p_{row}) + g \nabla (a \phi S_s p_s) \cdot \nabla D - q \alpha$

  - $\frac{\partial}{\partial t} (\phi a b S_s) - \nabla \cdot (a \phi S_s p_s) + \nabla \cdot (a \phi S_s p_{row}) + g \nabla (a \phi S_s p_s) \cdot \nabla D + g \nabla (a \phi S_s p_{row}) \cdot \nabla D = (q_s + q_r) \alpha$

- Loop
- Convergence?
- Yes
  - Predicted $x_{satur} = \{p_s, S_s, m_t\}_{satur}$

Obtaining a Gradient from an Undersaturated Oil Reservoir Simulator

It is clear from Eq. (1.10) that it is necessary to calculate $\frac{\partial f}{\partial x_i}$ and $\frac{\partial f}{\partial x_i}$ from

\[
\begin{align*}
  f_i (x_{a,s}, x_i, u_i) &= \left[ f_1 (p_s, S_s, m_t, p_s, S_s, m_t, q_s, q_s, q_s) \right] \\
  f_i (x_{a,s}, x_i, u_i) &= \left[ f_2 (p_s, S_s, m_t, p_s, S_s, m_t, q_s, q_s, q_s) \right] \\
  f_i (x_{a,s}, x_i, u_i) &= \left[ f_3 (p_s, S_s, m_t, p_s, S_s, m_t, q_s, q_s, q_s) \right]
\end{align*}
\]

\[
\begin{align*}
  f_1 (x_{a,s}, x_i, u_i) &= \frac{\phi a b S_s}{t_{s+1} - t_s} - \nabla \cdot (a \phi S_s p_s) + \nabla \cdot (a \phi S_s p_{row}) + g \nabla (a \phi S_s p_s) \cdot \nabla D - q \alpha \\
  f_2 (x_{a,s}, x_i, u_i) &= \frac{\phi a b S_s}{t_{s+1} - t_s} - \nabla \cdot (a \phi S_s p_s) + \nabla \cdot (a \phi S_s p_{row}) + g \nabla (a \phi S_s p_s) \cdot \nabla D - q \alpha \\
  f_3 (x_{a,s}, x_i, u_i) &= \frac{\phi a b S_s}{t_{s+1} - t_s} + \frac{\phi a b (1 - S_s)}{t_{s+1} - t_s} - \nabla \cdot (a \phi S_s p_s) + \nabla \cdot (a \phi S_s p_{row}) + g \nabla (a \phi S_s p_s) \cdot \nabla D - q \alpha
\end{align*}
\]
Calculating the former is easy, because an implicit discretization scheme was used. To calculate the latter, it is required to calculate $\frac{\partial F}{\partial p_s}$, $\frac{\partial F}{\partial s_s}$ and $\frac{\partial F}{\partial m_s}$. This could only be done if the "if-then-else" statement from figure Fig. 5 were differentiable, which of course is not the case. Differentiability can be forced in the following way:

First realize that the separate equations in the "if-then-else" statement of figure Fig. 5 can be combined in

$$r_s = \min \left\{ \frac{m_s}{1-S_u}, R''(p_s) b''_s (p_s) \right\}$$  \hspace{1cm} (1.37)

$$S_s = f^{-1}(m_s) \quad , \quad f(s) = b_s (p_s + p_{co}(s)) s + r_s (1-S_u - s)$$  \hspace{1cm} (1.38)

$$p_s = p_s - p_{co}(S_s) \quad , \quad p_s = p_s + p_{co}(S_s)$$  \hspace{1cm} (1.39)

$$p_s = f^{-1}(r_s) \quad , \quad f(p) = R''(p) b''_s (p) e^{-p}$$  \hspace{1cm} (1.40)

$$b_s = b_s (p_s) \quad , \quad b_s = b_s (p_s) e^{(b_s - p_s)} \quad , \quad b_s = b_s (p_s)$$  \hspace{1cm} (1.41)

$$R = R''(p_s)$$  \hspace{1cm} (1.42)

Forcing differentiability then comes down to finding appropriate regularized approximates of the min-function and consequently the "if-then-else" function

$$\min \{a, b\} = b + (a - b) s (b - a) = a + (b - a) s (a - b)$$  \hspace{1cm} (1.43)

with

$$s(x) = \begin{cases} 0 & \text{if } x < 0 \\ 1 & \text{if } x > 0 \end{cases}$$  \hspace{1cm} (1.44)

Let

$$\min_s \{a, b\} = b + (a - b) s_s (b - a) = a + (b - a) s_s (a - b)$$  \hspace{1cm} (1.45)

with $s_s(x) = \frac{1}{2} + \frac{1}{\pi} \arctan \left( \frac{\pi x}{\varepsilon} \right)$, then $\frac{\partial s_s}{\partial x}(0) = \frac{1}{\varepsilon}$ and $\lim_{x \to 0} s_s(x) = s(x)$

Performing these regularized differentiations on equations Eq. (1.37) through Eq. (1.43) results in

$$\frac{\partial r_s}{\partial p_s} = \frac{\partial}{\partial p_s} \left( R''(p_s) b''_s (p_s) + R''(p_s) b''_s (p_s) \right)$$

$$\frac{\partial r_s}{\partial S_u} = \frac{m_s}{1-S_u} R''(p_s) b''_s (p_s)$$

$$\frac{\partial r_s}{\partial m_s} = \frac{m_s}{1-S_u} R''(p_s) b''_s (p_s)$$

$$\frac{\partial S_s}{\partial p_s} = \frac{S_s b_s'(p_s + p_{co}(S_s)) (1 - S_u - S_s) + S_s b_s'(p_s + p_{co}(S_s)) - r_s}{m_s}$$

$$\frac{\partial S_s}{\partial S_u} = \frac{S_s b_s'(p_s + p_{co}(S_s)) (1 - S_u - S_s) + S_s b_s'(p_s + p_{co}(S_s)) - r_s}{m_s}$$

$$\frac{\partial S_s}{\partial m_s} = \frac{S_s b_s'(p_s + p_{co}(S_s)) (1 - S_u - S_s) + S_s b_s'(p_s + p_{co}(S_s)) - r_s}{m_s}$$

$$\frac{\partial r_s}{\partial p_s} = \frac{\partial r_s}{\partial S_u} = \frac{\partial r_s}{\partial m_s} = 0$$
\[ \frac{\partial p_e}{\partial p_e} = 1 + p_{\text{eq}}'(S_e) \frac{\partial S_e}{\partial p_e}, \quad \frac{\partial p_e}{\partial S_e} = p_{\text{eq}}'(S_e) \frac{\partial S_e}{\partial p_e}, \quad \frac{\partial p_e}{\partial m_e} = p_{\text{eq}}'(S_e) \frac{\partial S_e}{\partial m_e} \]

\[ \frac{\partial p_s}{\partial p_e} = \frac{\left( \frac{\partial p_s}{\partial p_e} - c r \right) e^{-c(\eta - \eta)} }{R_{\text{eq}}(p_s) b_{\text{eq}}(p_s) + R_{\text{eq}}(p_s) b_{\text{eq}}(p_s) - c R_{\text{eq}}(p_s) b_{\text{eq}}(p_s) c_s} \]

\[ \frac{\partial p_s}{\partial S_s} = \frac{\left( \frac{\partial p_s}{\partial S_s} - c r \right) e^{-c(\eta - \eta)} }{R_{\text{eq}}(p_s) b_{\text{eq}}(p_s) + R_{\text{eq}}(p_s) b_{\text{eq}}(p_s) - c R_{\text{eq}}(p_s) b_{\text{eq}}(p_s) c_s} \]

\[ \frac{\partial p_s}{\partial m_s} = \frac{\left( \frac{\partial p_s}{\partial m_s} - c r \right) e^{-c(\eta - \eta)} }{R_{\text{eq}}(p_s) b_{\text{eq}}(p_s) + R_{\text{eq}}(p_s) b_{\text{eq}}(p_s) - c R_{\text{eq}}(p_s) b_{\text{eq}}(p_s) c_s} \]

\[ \frac{\partial h}{\partial p_e} = b_{\text{eq}}'(p_e) \frac{\partial p_e}{\partial p_e}, \quad \frac{\partial h}{\partial S_s} = b_{\text{eq}}'(p_s) \frac{\partial p_s}{\partial S_s}, \quad \frac{\partial h}{\partial m_m} = b_{\text{eq}}'(p_m) \frac{\partial p_m}{\partial m_m} \]

\[ \frac{\partial b}{\partial p_e} = b_{\text{eq}}'(p_e) \frac{\partial p_e}{\partial p_e} \left( b_{\text{eq}}'(p_s) - b_{\text{eq}}'(p_m) c_s \right) e^{-c(\eta - \eta)} \]

\[ \frac{\partial b}{\partial S_s} = b_{\text{eq}}'(p_s) \frac{\partial p_s}{\partial S_s} \left( b_{\text{eq}}'(p_s) - b_{\text{eq}}'(p_m) c_s \right) e^{-c(\eta - \eta)} \]

\[ \frac{\partial b}{\partial m_m} = b_{\text{eq}}'(p_m) \frac{\partial p_m}{\partial m_m} \left( b_{\text{eq}}'(p_s) - b_{\text{eq}}'(p_m) c_s \right) e^{-c(\eta - \eta)} \]

\[ \frac{\partial R}{\partial p_e} = R_{\text{eq}}(p_e) \frac{\partial p_e}{\partial p_e}, \quad \frac{\partial R}{\partial S_s} = R_{\text{eq}}(p_s) \frac{\partial p_s}{\partial S_s}, \quad \frac{\partial R}{\partial m_m} = R_{\text{eq}}(p_m) \frac{\partial p_m}{\partial m_m} \]

**Speeding up the Forward Reservoir Simulator with Gradients**

In figure 5, it is described how \( x_{ini} \) can be calculated implicitly from Eq. (1.33) by Picard iterations: first the reservoir states are assumed to be known so all parameters can be calculated, second these parameters are used to setup PDE’s to update the states.

Since now the Jacobian \( \frac{\partial F(x_{ini}, x_{ini}, u_{ini})}{\partial x_{ini}} \) has become available, it is possible to replace the Picard iterations by much fewer Newton iterations. In that case \( x_{ini} \) is initialized by \( x_{ini}^0 = x_{ini} \) and updated by solving \( x_{ini}^{k+1} \) from

\[ \frac{\partial F}{\partial x_{ini}}(x_{ini}, x_{ini}, u_{ini})(x_{ini} - x_{ini}^{k+1}) = f(x_{ini}, x_{ini}, u_{ini}) \]

or symbolically \( x_{ini}^{k+1} = x_{ini} - \left( \frac{\partial F}{\partial x_{ini}}(x_{ini}, x_{ini}, u_{ini}) \right)^{-1} f(x_{ini}, x_{ini}, u_{ini}) \).

**Conclusions**

An adjoint simulator is very useful in history matching and optimal control problems. The adjoint requires the computation of the derivatives of the forward model. Reservoir simulators often utilize “if-statements” to distinguish between various situations that may occur during a production process, such as 2-phase versus 3-phase. The “if-statements” do not allow a direct derivative and have to be dealt with differently. In this paper it was shown how this can be done in the case of a (under)saturated oil. The interest in this case derived from the fact that significant pressure fluctuations may arise from dynamic control of a waterflood using smart wells. Failure to take the undersaturation into account leads to sub-optimal control strategies.

**References**


[6] Markovinovic, R. et al., Reduced Reservoir Models: Extending POD to include more general optimality conditions, ECMOR IX, Cannes, France, 2004